uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías Industriales
2017/2018

Trabajo Fin de Grado

Estudio de gestores de referencias bibliográficas y conectividad web con la plataforma Mendeley

Mireya Cestero de Dompablo

Tutor
Juan Carlos González Víctores

18 de Octubre, 2018, Leganés

AGRADECIMIENTOS

Una vez finalizado el proyecto, me gustaría dedicar unas palabras de agradecimiento a aquellas personas que me han acompañado durante este tiempo.

En primer lugar, debo agradecer el tiempo, esfuerzo, dedicación, paciencia y cercanía con la que mi tutor, Juan G. Victores, ha llevado este trabajo de fin de grado. Su excelente orientación y consejos han conseguido aumentar mi interés por el mundo de la informática y que disfrutara mientras realizaba este proyecto.

También quisiera agradecer a la Universidad Carlos III de Madrid y a todos sus profesores estos tres años. Han conseguido mantener viva mi fascinación por la ingeniería así como afianzar mis hábitos de trabajo. No debo olvidar mencionar a los profesores de la Escuela Politécnica de Breslavia (Polonia), por acogerme este último año y continuar estimulando mis ambiciones profesionales.

Por último, desearía agradecer a mi familia su cariño y ánimo que siempre me empuja a seguir hacia delante y mejorar. También a mis amigos del colegio, universidad y compañeros de Erasmus, por su ayuda y las experiencias compartidas.

RESUMEN

El presente trabajo consiste en explorar el funcionamiento de varios gestores de referencias. Asimismo, se estudia el concepto de tecnología web para conectar con un gestor de referencias bibliográficas: Mendeley.

La motivación principal es iniciar un proyecto ambicioso: poder gestionar archivos en varios gestores de referencias desde una única plataforma. En otras palabras, poder compartir instantáneamente un documento en varios gestores para ahorrar tiempo. Con esta motivación en mente, se decidió empezar por algo más sencillo y estudiar la viabilidad del proyecto al tratar de acceder a la cuenta de Mendeley y gestionar archivos desde un código en Python.

En esta memoria se abordarán los temas legales que envuelven al trabajo; es decir, a la protección de datos. Asimismo se hablará del entorno socio-económico del trabajo. Se pondrá en contexto el trabajo desarrollando los conceptos de gestores de referencias bibliográficas y tecnologías web. Finalmente, se explicará cómo se ha llevado a cabo la implementación del proyecto y cuál es el resultado obtenido.

ABSTRACT

This work consists in exploring the functioning of several reference managers. Also, the concept of web technology will be studied in order to connect with a bibliographic reference manager named Mendeley.

The main aspiration of the project is to start a more ambitious one: to be able to manage files in several reference managers from a single platform. In other words, the aim is to instantly share a document in several managers to save time. With this motivation in mind, it was decided to begin with something less complex and study the feasibility of the project by attempting to access Mendeley's account and managing files from a Python code.

In this report, legal issues that involve work will be addressed; that is, data protection. The social an economic aspects of the project will also be discussed. The work will be put in context by explaining the concepts of web technology and bibliographic reference managers. Finally, it will be explained how the implementation of the project was carried out and what are the results obtained.

Índice general

Índice de figuras

Índice de tablas

Glosario Inglés-Español

A

API: Application Programming Interface
Interfaz de Programación de Aplicaciones

Access token
Ficha de acceso

Access Token URL
Dirección URL de la ficha de acceso

All Documents
Todos los documentos

Authentication
Autenticación

Auth URL
Dirección URL de autenticación

Authorization
Autorización

Authorization code
Código de autorización

B

Bad Request
Mala petición

Basic
Básico

Bearer
Portador

C

Callback URL
Dirección URL de regreso

Client ID
Identificador del cliente

Client authentication
Autenticación del cliente

Client credentials
Credenciales de cliente

Client secret
Secreto de cliente

Consumer key
Clave de usuario

Copy
Copiar

Created
Creado

D

Data
Datos

Digest authentication
Autenticación de resumen

E

Endpoint
Extremo o punta

Export with Annotations
Exportar con anotaciones

F

File
Archivo

Forbidden
Prohibido

G

Grant Type
Tipo de concesión

H

HTTP: HyperText Transfer Protocol
Protocolo de transferencia de hipertexto

I

Implicit grant
Concesión implícita

Insert citation/bibliography
Insertar cita/bibliografía

Install
Instalar

J

JSON: JavaScript Object Notation
Notación de Objetos de JavaScript

L

Label (for)
Etiqueta (para)

N

New project
Nuevo proyecto

No Content
Sin contenido

Not Found
No se encuentra

O

OAuth info
Información de autenticación

P

Params
Parámetros

Password
Contraseña

Q

Querystring
Cadena de consulta

R

REST: REpresentational State Transfer
Transferencia de estado representacional

Request
Petición

Request body
Cuerpo de la petición

Request header
Encabezado de la petición

Request Timeout
Tiempo de espera de petición agotado

Resource owner password credentials
Credenciales de contraseña del dueño de los recursos

Response body
Cuerpo de la respuesta

Response status-code
Código de estado de la respuesta

Run
Ejectutar, correr, operar

S

SDK: Software Developer Kit
Kit de desarrollo de software

SOAP: Simple Object Access Protocol
Protocolo de acceso simple a objetos

Scope
Ámbito

State
Estado

String
Cadena o línea de texto

T

Tools
Herramientas

U

URL: Uniform Resource Locator
Localizador de recursos uniforme

Unauthorized
Sin Autorización

Unprocessable Entity
Entidad no procesable

Username
Nombre de usuario

X

XML: eXtensible Markup Language
Lenguaje de marcas extensible

Capítulo 1

Introducción

En este capítulo se describirá la motivación del trabajo así como las alternativas de diseño encontradas. Asimismo, se justificará la decisión tomada. También se explicará la planificación requerida para llevarla a cabo. Por último, se hará un breve resumen del contenido de cada capítulo de la memoria.

1.1. Planteamiento del problema

Este trabajo forma parte de un proyecto muy ambicioso y de posible impacto a nivel internacional en el ámbito científico; especialmente, para los investigadores. La motivación, en pocas palabras, es la siguiente: automatizar la gestión de documentos científicos y referencias bibliográficas en varias plataformas para realizarlo desde una única aplicación. A continuación, se explicará en más detalle esta propuesta.

Hoy en día, la información puede ser accedida desde numerosos dispositivos como ordenador portátil, teléfono móvil o tablet con pocos clics. Por otro lado, las redes sociales juegan un significativo papel en todos los ámbitos. Esto se debe a que permiten que los usuarios se conecten entre sí desde cualquier lugar y compartan información. La idea de este proyecto es aunar los conceptos previamente mencionados.

Por ejemplo, existe una popular red social para compartir imágenes y vídeos llamada Instagram. Por otro lado, Facebook, la conocida red social creada por Mark Zuckerberg, es muy utilizada en la sociedad para compartir fotos, textos breves o enlaces, entre otros. Existe una funcionalidad en Instagram que consiste en, con tan solo seleccionar el botón "Compartir en Facebook", subir una imagen a Instagram y a Facebook al mismo tiempo. Esto elimina la necesidad de realizar el tedioso proceso de: salir de la aplicación Instagram, entrar en la aplicación de Facebook, encontrar el botón para subir fotos en Facebook, elegir la foto y, finalmente, compartirla en el muro de Facebook.

El sencillo ejemplo explicado anteriormente refleja la idea de este proyecto. Sin embargo, en este caso, en lugar de subir fotos, el propósito es compartir e incluso gestionar (descargar un listado, eliminar y modificar) documentos científicos y referencias bibliográficas. Igualmente, no se pretende compartir documentos científicos en redes sociales como Facebook. El objetivo del proyecto es hacerlo en gestores de referencias bibliográficas y documentos, que se explicarán más adelante, como Mendeley, ORCID, ResearchGate, etc. para que los usuarios los compartan y gestionen como deseen.

En resumen, la meta final es la siguiente: desde una sola plataforma poder compartir en otras plataformas (Mendeley, ResearchGate, etc.), modificar y gestionar referencias bibliográficas y documentos científicos. Este trabajo se centra en conseguir interactuar con la plataforma Mendeley sin necesidad de acceder a ella directamente.

1.2. Alternativas de diseño

En esta sección, se describirán las alternativas de diseño contempladas para llevar a cabo el trabajo. La principal disyuntiva a la hora de empezar el trabajo era el modo en que obtener y compartir información en Mendeley con un lenguaje de programación de propósito general (p. ej. Python). A continuación se exponen las dos opciones valoradas.

a) Automatizando el uso del navegador

   Un modo de acceder al contenido de una página web sin "entrar" en ella directamente, es conseguir pulsar botones y rellenar campos en un navegador desde un código. En el apartado 4.1 se explicará con detalle cuál es el procedimiento. El problema es que este método depende de la interfaz; si ésta cambia, este método puede fallar.

b) A través de la API de Mendeley

   Como se explicará posteriormente en el apartado 3.1.1, una API se emplea para intercambiar información con una página web a través de un código. Es un método muy extendido en la comunidad informática y que será de gran relevancia para este trabajo.

Ambos métodos poseen ventajas e inconvenientes y es fundamental considerar ambos. En primer lugar, el método a) resulta muy simple de implementar desde Python puesto que existen módulos como *Selenium* que lo facilitan. Por otro lado, si cambia la interfaz, este método puede dejar de ser útil a menos que se actualice. Algunas páginas web cambian su interfaz más a menudo que otras y eso supone un gran problema para este método.

Por otro lado, intercambiar información a través de *requests* o peticiones con la API de Mendeley, opción b), es el modo más elegante. La interacción con la API elimina el problema de las modificaciones de la interfaz puesto que no afectan. No obstante, las opciones que ofrece una API, en ocasiones, son limitadas y dificultan acceder o compartir cierta información de determinada forma.

1.2.1. Justificación de la solución, objetivos y planificación

Tras contemplar las opciones propuestas en el apartado anterior y valorar sus ventajas e inconvenientes, se decidió emplear ambas. A continuación se justificará dicha decisión y se describirá la planificación que ha sido necesaria para llevara cabo los objetivos del trabajo.

Justificación de la solución

Como se verá en el apartado 4.1, resulta muy cómodo obtener los datos personales de identificación (o credenciales) para registrarse en una página web, empleando el método a), automatizando el uso del navegador. Además, es una página cuya interfaz no se espera que cambie. Este procedimiento no se realiza a través de la API puesto que la documentación de la misma no especifica cómo; sino que usa las credenciales directamente.

Una vez obtenidos las credenciales, la documentación de la API de Mendeley sí que permite descargar listas de archivos, borrar y subir documentos de manera relativamente sencilla. Es por eso que se optó por el método b), interactuar con Mendeley a través de la API para intercambiar información. Este proceso se detallará en el apartado 4.2.

Objetivos y planificación del proyecto

El objetivo de este trabajo es iniciar el proyecto descrito en el apartado 1.1: combinar gestores de referencias en una única plataforma para gestionar documentos científicos y referencias bibliográficas. El objetivo es ahorrar tiempo y aportar comodidad a los científicos al compartir y gestionar documentos de investigación.

En concreto, este TFG (Trabajo de Fin de Grado) se encarga de investigar e implementar un modo de conectar con la plataforma Mendeley mediante el lenguaje de programación Python. Para ello será necesario la realización de las siguientes fases:

1. Estudio de gestores de referencias y tecnologías web

   Estudiar los gestores bibliográficos (funcionamiento de Mendeley, ResearchGate, etc.) así como tecnologías web (API, OAuth 2.0, etc.) gracias a las cuales poder interactuar con dichos gestores a través del lenguaje de programación Python.

2. Conseguir las credenciales de manera automática

   Obtener las credenciales necesarias (*access token*) para realizar peticiones, mediante Python, a la API de Mendeley de una manera automática; es decir, sin acceder a la página que los proporciona directamente.

3. Obtener, subir y borrar publicaciones de Mendeley

   A través de la API, conseguir descargar listas de publicaciones, acceder a una en concreto, subir archivos y borrar entradas de Mendeley.

4. Revisar y redactar

   Una vez finalizado el proyecto, revisarlo y organizar el código final. Después, redactar la memoria y, para finalizar, revisión final por parte del tutor.

A continuación se mostrará una diagrama de Gantt (tabla 1.1) con la descripción detallada de la planificación expuesta previamente.

Tabla 1.1: Diagrama de Gantt: planificación del proyecto

		PLANIFICACIÓN DEL PROYECTO								
Nº	**NOMBRE DE LA TAREA**	**15may-5jun**	**6jun-26jun**	**27jun-3jul**	**4jul-11jul**	**12jul-16jul**	**17jul-26jul**	**27jul-28jul**	**29jul-31ago**	**1sept-19sept**
1	**Estudio de gestores de referencias y tecnologías web**									
1.1	Estudio de gestores de referencias									
1.2	Estudio de tecnologías web									
2	**Obtener el *access token* de manera automática**									
2.1	Obtener el *access token*									
2.2	Automatizar el proceso									
3	**Obtener, subir y borrar documentos de Mendeley**									
3.1	Obtener documentos									
3.2	Subir documentos									
3.3	Eliminar documentos									
4	**Revisar y redactar**									
4.1	Revisar y organizar el código									
4.2	Redactar									
4.3	Revisión final por parte del tutor									

1.3. Estructura del documento

En este apartado se describirá brevemente el contenido de cada uno de los seis capítulos que componen este documento; incluyendo el capítulo presente (*Introducción*). A parte de los seis capítulos principales, la memoria cuenta con un glosario de traducciones inglés-español y una bibliografía en la que encontrar las referencias bibliográficas en las que se basa este documento.

- Capítulo primero: Introducción

  En este capítulo se describe el planteamiento del problema, las alternativas de diseño a la hora de implementarlo y se justifica la decisión tomada. Asimismo, se realiza un resumen del contenido la memoria por capítulos.

- Capítulo segundo: Ámbito legal y socio-económico

  En este capítulo se hablará del entorno socio-económico que envuelve a los pilares del trabajo (gestores de referencias y tecnologías web); así como del impacto que se espera obtener y su viabilidad. También se expondrá el marco regulador centrado en la protección de datos y se realizará el cálculo estimado del presupuesto.

- Capítulo tercero: Estado del arte

  Este capítulo, de carácter teórico, se centra en contextualizar el proyecto. Para ello, explica los conceptos de gestor de referencias y tecnologías web (REST API, OAuth 2.0, etc.) para intercambiar información con ellos a través del lenguaje de programación Python. Se hablará de tres gestores de referencias: ResearchGate, Google Scholar y Mendeley, y se estudiará si estos disponen de una API.

- Capítulo cuarto: Implementación

  En este capítulo se pone en práctica lo explicado previamente acerca de tecnologías web para interactuar con la plataforma Mendeley. Esto se lleva a cabo a través del lenguaje de programación Python y consta de dos bloques: registrarse (acceder a una cuenta) y gestionar publicaciones (descargar un listado, subir y eliminar).

- Capítulo quinto: Guía de uso

  Este capítulo está enfocado en explicar a un usuario qué requisitos previos necesita, cómo debe de usarse el programa y qué opciones ofrece.

- Capítulo sexto: Conclusiones

  En el capítulo final se resume lo expuesto en la memoria, se analizan los resultados obtenidos y se describen los planes de futuro para este trabajo.

Capítulo 2

Ámbito legal y socio-económico

Este capítulo se ocupa de comentar los aspectos tanto legales como socio-económicos relativos a este TFG (Trabajo de Fin de Grado). Se hablará por tanto de las leyes y normativas aplicables al trabajo así como del entorno socio-económico y su posible impacto. Asimismo, se realizará un estudio del presupuesto estimado de éste.

2.1. Marco regulador

En este apartado se procederá a describir las leyes, reglamentos y directivas aplicables a este trabajo en orden cronológico. Dado que este TFG lidia con la gestión de archivos y obtención y uso de datos personales por métodos electrónicos, resulta oportuno estudiar las leyes relacionadas con los derechos fundamentales y la protección de datos haciendo hincapié en los aspectos electrónicos [1]. Es preciso aclarar que las normativas principales y vigentes son: el Artículo 18.4 de la Constitución, la Ley Orgánica (LO) 15/1999; la más relevante en la materia, el Reglamento Europeo (UE) 2016/679 y el reciente Reglamento General de Protección de Datos de la Unión Europea.

2.1.1. Derechos fundamentales

En este apartado se describirán las regulaciones relativas a la protección de datos y los derechos fundamentales tanto en el ámbito europeo como en el español.

- **Derecho Constitucional Político Español según la Constitución de 1978 - II Derechos Fundamentales y Órganos del Estado**

  En este documento se indica que la LO 5/1992 del 29 de octubre regula el procesamiento automático de datos personales. Esta ley fue sustituida posteriormente por la LO 15/1993, del 13 de diciembre que, como se explica en el apartado 2.1.2, regula la protección de datos de carácter personal [2].

- **Carta de Derechos Fundamentales de la Unión Europea, de 18 de diciembre del 2000**

  En el artículo octavo se establece que los datos personales que conciernan a una persona han de ser usados, con el consentimiento dicha persona afectada y de acuerdo a un fin concreto y leal. Asimismo, toda persona puede acceder a los datos que le han sido recogidos y rectificarlos si desea [3] [4].

2.1.2. Protección de datos personales

En este apartado se comentará la regulación relativa a la protección de datos en Europa y en España.

- **Directiva 95/46/CE del Parlamento Europeo y del Consejo, 24 octubre 1995**

  Esta normativa, derogada, aborda el tema de la protección de datos personales al establecer que los estados miembros pueden limitar el derecho a la protección de datos siempre y cuando ésta se base en proteger la seguridad pública o el Estado, por ejemplo [1].

- **Ley Orgánica 15/1999, de 13 de diciembre de 1999, de protección de datos de carácter personal**

  Esta ley orgánica está precedida por la LO 5/1992, del 29 de octubre de regulación del tratamiento automático de los datos de carácter personal (LORTAD) [5]. Tal y como se describe en [6], esta ley defiende y protege las libertades y derechos de las personas físicas en lo relativo a los datos personales (artículo primero). Se aplica, por tanto, a aquellos datos personales susceptibles de ser tratados o usados por sectores tanto privados como públicos (artículo segundo). Esta ley también regula que, al recoger datos, se informe expresamente del motivo de su recogida y se atienda al objetivo de la misma exclusivamente (artículos cuarto y quinto). Asimismo, con carácter general, debe existir consentimiento por parte de la persona que entrega los datos (artículo sexto).

  Para garantizar la seguridad de los datos, se exige al responsable de un fichero o de su tratamiento, que tome las medias oportunas para evitar la pérdida, divulgación o alteración de los mismos (artículos noveno y décimo). En caso de comunicar los datos personales, se debe disponer del consentimiento del propietario de los datos. Asimismo el fin debe estar estrictamente relacionado con las funciones legítimas del cedente (artículo undécimo). Si es precisa la intervención de un tercero, el tratamiento debe estar regulado por un contrato y atenerse a él; tras dicho contrato los datos personales deben ser devueltos (artículo decimotercero) [6].

- **Reglamento Europeo (UE) 2016/679, de 27 de abril de 2016**

  Este reglamento establece en el artículo quinto que los datos han de ser tratados con transparencia, limitación y lealtad a su uso legítimo. El titular de los datos debe ser informado de todo lo relacionado en cuanto a la gestión de los mismos. Asimismo se indica que la circulación de datos no podrá ser restringida debido a la protección de personas físicas (artículo primero). Con este reglamento queda derogada la Directiva 95/46/CE [7] [8].

- **Reglamento General de Protección de Datos de la Unión Europea, de 25 de mayo de 2018**

  En diciembre de 2015, se firmó un acuerdo para la aprobación del futuro Reglamento General de Protección de Datos (Reglamento (UE) 2016/679), el cual deroga la Directiva 95/46/CE. Este reglamento entró en vigor el pasado mayo de 2018. Entre las novedades que aporta al marco legal de la protección de datos, destaca el mayor control que se les otorga a los usuarios sobre sus datos [9]. Regula también derechos al acceso y al olvido, relaciones responsable-encargado, medidas de responsabilidad activa, transferencias internacionales y tratamiento de datos de menores [1].

2.1.3. Protección de datos personales en relación a Internet

Este apartado se centra en la legislación relativa a la protección de datos de carácter personal en el ámbito informático en orden cronológico a nivel europeo y español.

- **Artículo 18.4 de la Constitución Española (1978)**

  Este artículo se inspira en las previsiones del Convenio 108 del Consejo de Europa, de 28 de enero de 1981. En él, se establece como derecho fundamental y autónomo el control del flujo de datos que concierte a la privacidad de una persona. [10].

- **Convenio nº 108 del Consejo de Europa, de 28 de enero de 1981, para la protección de datos con respecto al tratamiento automático**

  Si bien este convenio queda en desuso y relegado a la tecnología de la época [7], éste establece principios para la protección de datos como el derecho al olvido y la utilización no abusiva. Este convenio fue desarrollado, más tarde, en España inicialmente por la LO 5/1992 [5].

[1] Agencia Española de Protección de Datos, "Guía del Reglamento General de Protección de Datos para responsables de tratamiento". [En línea] Disponible en: `https://www.aepd.es/media/guias/guia-rgpd-para-responsables-de-tratamiento.pdf` (último acceso 23 de agosto de 2018)

- **Ley Orgánica 5/1992, de 29 de octubre de 1992, de regulación del tratamiento automatizado**

  Esta ley se centra en conceptos como el derecho a la intimidad y privacidad informática. Indica que la ley deberá limitar el uso de la informática y garantizar así la intimidad y el honor de las personas [11].

- **Directiva 97/66/CE del Parlamento Europeo y del Consejo, de 15 de diciembre de 1997**

  Esta directiva, si bien ha sido derogada, establece el derecho la intimidad protegida en relación con el ámbito de la informática y las telecomunicaciones. Esta normativa actualiza la Directiva 95/46/CE y fue sustituida por la 02/58/CE [5].

- **Marco regulador de las comunicaciones electrónicas (nivel europeo) [1] [12] [2]:**

  - Directiva 2002/20/CE del Parlamento Europeo y del Consejo, de 7 de marzo de 2002, relativa a la autorización de redes y servicios de comunicaciones electrónicas (Directiva autorización).
  - Directiva 2002/19/CE del Parlamento Europeo y del Consejo, de 7 de marzo de 2002, relativa al acceso a las redes de comunicaciones electrónicas y recursos asociados, y de su interconexión (Directiva acceso).
  - Directiva 2002/22/CE del Parlamento Europeo y del Consejo, de 7 de marzo de 2002, relativa al servicio universal y los derechos de los usuarios en relación con las redes y los servicios de comunicaciones electrónicas (Directiva servicio universal).
  - Directiva 2002/58/CE del Parlamento Europeo y del Consejo, de 12 de julio de 2002, relativa al tratamiento de los datos personales y a la protección de la intimidad en el sector de las comunicaciones electrónicas (Directiva sobre la privacidad y las comunicaciones electrónicas).
  - Reglamento Nº 1211/2009 del Organismo de Reguladores Europeos de las comunicaciones electrónicas (ORECE), de 25 de noviembre de 2009.
  - Reglamento Nº 531/2012 del Parlamento Europeo del Consejo, de 13 de junio de 2012, relativo a la intinerancia de las redes públicas de comunicaciones móviles en la Unión Europea.

[2] *EUR-Lex - l24216a - EN - EUR-Lex.* [En línea] Disponible en `https://eur-lex.europa.eu/homepage.html` (último acceso 26 de agosto de 2018)

Si bien no se trata de una ley, es relevante destacar la existencia del conocido **Grupo de Trabajo del Artículo 29** sobre Protección de las Personas en lo que respecta al Tratamiento de los Datos Personales. Este organismo opera a nivel de la Unión Europea reuniendo a las autoridades en la materia de los países miembros. Surge con el fin de aunar lo establecido en el Convenio nº 108 del Consejo de la UE y en el artículo 29 de la Directiva 95/46/CE [13] [14].

España, como estado miembro de la Unión Europea, dispone de un organismo encargado de regular la protección de datos personales: **Agencia Española de Protección de Datos, AEPD.** Aunque España no se adaptó a lo establecido en el convenio Nº 108 del Consejo de Europa rápidamente, en la actualidad es un Estado con normas muy restrictivas [15][16]. Es relevante destacar que la legislación de protección de datos por la que se rige la AEPD afecta tanto a soportes digitales como en papel.

2.2. Entorno socio-económico

Esta sección está centrada en torno al estudio de los aspectos sociales y económicos tanto de las tecnologías web como los gestores de referencias. Asimismo, en el segundo apartado de esta sección, se expondrá la viabilidad del proyecto y el impacto que se espera obtener del mismo.

2.2.1. Estudio del entorno socio-económico de las tecnologías web y los gestores de referencias

En esta sección se procederá a comentar los aspectos sociales y económicos de los dos conceptos relevantes para este TFG: tecnologías web (ver apartado 3.1 para más detalles) y gestores de referencias (ver apartado 3.2 para más información).

Entorno social

Este apartado se ha dedicado a los aspectos sociales de este proyecto. Como se menciona en la introducción de este apartado, se estudiará el entorno social de las tecnologías web y, después, de los gestores de referencias.

- **Entorno social de tecnologías web**

  Como se definirá posteriormente en el apartado 3.1, las tecnologías web hacen referencia a la implementación técnica de todo aquello relacionado con la interconexión de ordenadores para el intercambio de información. La tecnología web que destaca en este trabajo es una API (ver apartado 3.1.1). Estas permiten a un desarrollador acceder al contenido de una página e interactuar con ella siguiendo unos protocolos de autenticación. Los protocolos de autenticación se emplean para verificar que el usuario es quien dice ser y tiene acceso a determinado contenido.

  Numerosas páginas web y redes sociales hoy en día disponen de una API. Gracias a esa API, otras páginas o redes sociales pueden iteractuar con ella, obteniendo datos como el número de visitas. Con dichos datos se pueden obtener estadísticas sociales y realizar clasificaciones que ayudan a una página o empresa a la hora de promocionarse en los diferentes sectores. Por otro lado, se puede interpretar a las tecnologías web como herramientas social en las que se basan múltiples plataformas para asociar cuentas de redes sociales a dichas plataformas. Por ejemplo, gracias a una API, una página web en la que aparece un artículo puede ofrecer al usuario la opción de compartir dicho artículo directamente en una red social, como Twitter o Facebook, con tan solo validar su identidad o escribiendo sus credenciales.

- **Entorno social de los gestores de referencias bibliográficas**

  Un considerable número de gestores de referencias bibliográficos como Mendeley o ResearchGate cuentan con un aspecto de red social. Es decir, un usuario puede crearse un perfil en uno de los previamente mencionados gestores de referencias, y compartir publicaciones con otros usuarios. En ocasiones, también puede enviarles mensajes o notificaciones si se ha citado un artículo, por ejemplo. Generalmente, se permite hacerse seguidor de otros usuarios para mantenerse al día de las novedades en el mundo de la investigación.

  ResearchGate, por ejemplo, también permite enviar mensajes, hacer preguntas y responderlas, añadir una descripción del usuario y seguir a otros. Estos aspectos de red social que muchos gestores de referencias han adoptado, los hacen más modernos y atractivos a la sociedad en la actualidad. Hoy en día es extremadamente inusual que una persona no disponga de, por lo menos, un perfil en una red social. Algo similar ocurre con los gestores de referencias. Con el fin de agilizar las investigaciones y divulgaciones científicas, los usuarios se crean perfiles para compartir sus publicaciones y acceder a las de otros.

Entorno económico

En este apartado se comentará brevemente el entorno económico de los dos pilares de este trabajo. Dichos pilares son: tecnologías web y gestores de referencias bibliográficas.

- **Entorno económico de tecnologías web**

  Como se menciona en el apartado anterior, el uso de una API puede beneficiar a una página, red social o empresa gracias a la obtención de datos para su posterior transformación en estadísticas, por ejemplo. Las estadísticas hacen posible a las empresas distinguir nichos de mercado y crear soluciones para solventarlos. También permiten, simplemente, decidir en qué, cómo, dónde y cuándo centrar su trabajo o su inversión. A esta ventaja de las APIs, se le debe añadir otra de las posibilidades que ésta ofrece: crear aplicaciones de escritorio o móvil para que el usuario no necesite entrar en un navegador para acceder a una plataforma.

- **Entorno económico de gestores de referencias bibliográficas**

  Si bien el ámbito social de los gestores de referencias bibliográficas con aspectos de red social (p. ej. Mendeley) tiene gran relevancia; ese no es el caso del aspecto económico de los mismos. Esto se debe a que su función última es gestionar publicaciones y perfiles de usuarios para que se conecten entre sí. Las publicaciones que se comparten son de carácter científico e investigación; lo que sugiere que, en un futuro, sí podrán impactar en la economía. No obstante, los gestores de referencias como tal no cuentan con un entorno económico relevante.

2.2.2. Viabilidad e impacto socio-económico del proyecto

A continuación, se estudiará la viabilidad del trabajo presentado en este TFG. Del mismo modo, se comentará el impacto que se espera obtener como resultado tras la implementación completa del proyecto.

En el apartado 1.1 se plantea la idea del trabajo que se ha llevado acabo. Se menciona también que este trabajo forma parte de un proyecto mayor. Este TFG sirve de inicio de un proyecto muy relevante para, en pocas palabras, aunar y automatizar la gestión de publicaciones científicas en distintos gestores de referencias desde una única aplicación. Una vez aclarado esto, en este apartado se hablará de la viabilidad e impacto del proyecto final, con comentarios acerca de este trabajo en particular.

- **Viabilidad del proyecto**

  En cuanto a la viabilidad del proyecto, se puede adelantar que sí se trata de un proyecto que se puede llevar a cabo en la práctica. Durante su realización se deberá sobrepasar más o menos obstáculos dependiendo del gestor de referencias que se quiera integrar en la anteriormente mencionada aplicación.

  Por ejemplo, Mendeley ofrece una extensa documentación para interactuar con su API. No obstante, aún existen dificultades para integrar todas las funcionalidades de dicho gestor (ver apartado 4.2.3 en relación a subir referencias de un archivo BibTex), lo que supone un obstáculo. Por otro lado, gestores como ResearchGate u ORCID dificultan la implementación de la aplicación que combina gestores de referencias puesto que no ofrecen una API pública. En algunos casos es necesario obtener una cuenta especial o abonar una cantidad de dinero para acceder a las funcionalidades de la API.

  Dichos impedimentos pueden solucionarse de dos modos. Por un lado, en caso de que las plataformas hagan públicas y/o gratuitas las APIs, se trata, tan solo, de una cuestión de tiempo. Por otro lado, resulta una más aconsejable, aunque posiblemente más compleja opción, investigar métodos alternativos a las APIs para acceder a los datos de los gestores.

  Como comentario final, a la hora de implementar la aplicación que aúne gestores, será necesario verificar que no se incumpla ninguna ley (apartado 2.1) ni se viole las normas de privacidad de cada gestor. No obstante, se estima que el trabajo es viable a nivel legal.

- **Impacto del proyecto**

  El impacto esperado a la finalización del proyecto es de nivel internacional en el ámbito científico y de investigación. Se entiende por finalización del proyecto a la continuación de este trabajo, es decir, la implementación de una aplicación que aúne varios gestores de referencias.

  Dicha aplicación permitiría ahorrar tiempo a los usuarios que deseen compartir publicaciones en más de un gestor de referencias. Asimismo, supondría una gran comodidad para los científicos y agilizaría los procesos de investigación. Los usuarios podrán seguir compartiendo sus publicaciones y leyendo las de otros usuarios en los gestores de referencias que frecuenten. No obstante, lo realizarán con mayor comodidad y velocidad. Se espera por tanto que se fomente el interés por la ciencia, se agilice la investigación y se promueva el contacto social entre investigadores y científicos.

En particular, el impacto de este TFG en el momento de redacción de esta memoria (agosto de 2018) quizás pueda resultar poco significativo. No obstante, sí que es destacable puesto que se trata del primer paso de un proyecto de relevancia internacional. Este trabajo prueba que es posible gestionar publicaciones de una plataforma sin acceder a ella directamente. Asimismo, inicia el camino a recorrer para, en el día de mañana, agilizar la divulgación de documentos científicos.

2.3. Presupuesto del proyecto

En esta sección se realizará un cálculo estimado del presupuesto necesario para llevar a cabo este trabajo de fin de grado. No se ha llevado a cabo ninguna subcontratación de tareas; asimismo el trabajo funciona sin coste alguno. Los costes indirectos como dietas o viajes no son aplicables a la realización de este trabajo. Por lo tanto, los costes relevantes son los siguientes: costes materiales, costes de personal y costes totales.

Costes materiales

Dado que el presente trabajo es un proyecto exclusivamente de software, los costes materiales no resultan excesivos en caso de disponer de un ordenador y un escritorio. Los costes materiales que pueden ser requeridos para el trabajo son los siguientes:

- Ordenador: ordenador portátil u ordenador de sobremesa con la equipación necesaria (teclado, pantalla y ratón).
- Escritorio: silla y mesa de trabajo.
- Editor de Python 3.7: PyCharm 3.5 *Community Edition* es un editor gratuito del lenguaje de programación Python perteneciente a la compañía JetBrains y compatible con el lenguaje de programación Python versión 3.7.
- Editor de LaTeX: para la redacción de la presente memoria se ha optado por el formato de redacción LaTeX y el editor de LaTeX online llamado Overleaf, también sin coste alguno.
- Gastos de luz y acceso a Internet.

El único coste relevante es el del ordenador, puesto que los editores PyCharm y Overleaf son gratuitos y los gastos de luz e Internet resultan despreciables a la hora de calcular los costes materiales. La tabla 2.1 muestra el cálculo del coste imputable del ordenador teniendo en cuenta la depreciación (ver ecuación 2.1).

Tabla 2.1: Tabla de presupuesto de material: ordenador portátil Acer [3]

Descripción	**Coste (€)**	**% Uso dedicado**	**Dedicación (meses)**	**Periodo de depreciación (meses)**	**Coste imputable (€)**
Portátil Acer	1000	10	4	60	66,67

La ecuación 2.1 que aparece a continuación indica el cálculo de la amortización

$$A = \frac{B}{C} * D * E \tag{2.1}$$

Donde: A representa la amortización; B, el numero de meses desde la fecha de en que el equipo es utilizado; C, periodo de depreciación (60 meses); D, coste del equipo (sin IVA) y E, porcentaje del uso que se dedica al proyecto.

Costes de personal

En cuanto a los costes de personal, las dos únicas personas implicadas en este trabajo han sido el tutor, Juan Carlos González Víctores, encargado de dirigir y revisar el trabajo y la autora, Mireya Cestero de Dompablo, cuya misión fue llevar a cabo el trabajo y redactar la memoria del trabajo.

Puesto que el tutor del trabajo, Juan G. Victores, es doctor en Robótica e Inteligencia Artificial así como Investigador en *RoboticsLab* (UC3M - Universidad Carlos III de Madrid), el precio de sus horas de trabajo son tomadas como las de Ingeniero Senior (35 €/hora aproximadamente). Por otro lado, dado que la autora es estudiante del cuarto curso del Grado en Ingeniería en Tecnologías Industriales en la Universidad Carlos III de Madrid, se toman sus horas de trabajo como las de Ingeniero Junior (15 €/hora aproximadamente).

[3] Estructura de tabla obtenida en: asrob-uc3m, Redactar". [En línea] Disponible en: `https://github.com/asrob-uc3m/recursos-digitales/blob/master/plantillas/Formulario_PresupuestoPFC-TFG.xlsx` (último acceso 17 de septiembre de 2018)

La tabla 2.2 que aparece a continuación muestra un resumen de los costes de personal del trabajo. Se obtiene como coste de personal total estimado: 6550€.

Tabla 2.2: Tabla de presupuesto de personal [3]

Nombre y Apellidos	**Categoría**	**Precio (€/hora)**	**Horas de trabajo**	**Sueldo estimado (€)**
Juan Carlos González Víctores	Ingeniero Senior	35	50	1750
Mireya Cestero de Dompablo	Ingeniero Junior	15	320	4800
TOTAL				6550

Costes totales

La tabla 2.3 muestra el valor del presupuesto total estimado de este trabajo. Para ello se han tenido en cuenta los gastos de personal (tutor y autora del trabajo) así como los materiales imputables del ordenador.

Se obtiene como coste total del trabajo: 6616,67€.

Tabla 2.3: Presupuesto total aproximado del trabajo [3]

Tipo de coste	**Presupuesto de costes totales (€)**
Costes de personal	6550
Costes materiales	66,67
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes indirectos	0
TOTAL	6616,67

Capítulo 3

Estado del arte

En este capítulo se pondrá en contexto este TFG hablando de las tecnologías web y las APIs. Asimismo se explicará la manera de conseguir la autorización necesaria para realizar una petición a una página web a través de la API y gestionar información de dicha página. También se hará un análisis de tres de los gestores de referencias bibliográficas más empleados por investigadores (ResearchGate, Google Scholar y Mendeley); se estudiará tanto su funcionamiento como su API, en caso de existir.

3.1. Tecnologías Web

En este apartado se definirá brevemente el concepto de tecnologías web y se describirán las APIs y tipos de autorización que pueden requerir. También se explicará cómo se lleva a cabo la conexión con la API via *requests* en Python.

El término tecnología proviene de dos palabras griegas: *tekne* (ténica) y *logos* (conocimiento). Se define, por tanto, como la aplicación práctica de la ciencia enfocada a resolver problemas técnicos. Por otro lado, la palabra web, anglicismo que se traduce como red, hace referencia a las páginas web o, en un sentido más amplio, a Internet. Internet es una red internacional de ordenadores interconectados para proporcionar numerosos servicios [17]. Las páginas web se caracterizan por contener todo tipo de información como texto o sonido en una dirección de Internet concreta.

Por lo tanto, se entiende por tecnologías web todo aquello relacionado con la interconexión entre ordenadores con el objetivo de intercambiar información. Las APIs facilitan dicha conexión como se explica en el apartado 3.1.1; para ello, es necesario una autorización (véase apartado 3.1.2) y un lenguaje de programación para comunicarse con la API; por ejemplo, Python. En el apartado 3.1.3 se muestra como implementar peticiones a la API desde Python.

3.1.1. Introducción a las APIs

A continuación, se definirá el concepto de API, se hablará de los principales tipos y se describirá en más profundidad una REST API por su relevancia para este proyecto.

Las siglas API representan una Interfaz de Programación de Aplicaciones. Una explicación muy sencilla de su funcionalidad es la siguiente: actúan como un camarero en un restaurante. El camarero toma nota de lo que ordena el cliente (*request*). Nótese que el cliente está autorizado a sentarse en el restaurante y pedir una comida. El camarero entrega la petición de comida en la cocina (sistema del que se quiere obtener información). Una vez la comida ha sido preparada (se ha procesado la *request*), el camarero regresa a la mesa con la comida (información que el cliente deseaba obtener) [1].

Una API proporciona la posibilidad de acceder al contenido de la base de datos de una página web sin acceder a la misma directamente ni a su código fuente. Volviendo a la analogía del camarero presentada en el párrafo anterior; el cliente no tiene que ir a la cocina para pedir su comida ni tiene por qué saber cómo la preparan, simplemente recibe la comida preparada en su mesa[1]. Una API es una interfaz puesto que sirve de nexo para la comunicación entre dos entidades o aplicaciones gracias al código escrito por un programador para obtener o proporcionar determinada información.

Tipos de APIs

En este apartado se hablará de tres tipos de APIs muy relevantes para los desarrolladores; se distinguen entre si según si son basadas en bibliotecas, de funciones en sistemas operativos o de servicios web [2].

- APIs basadas en bibliotecas

  Este tipo de API facilita el intercambio de información mediante la importación de bibliotecas. Un claro ejemplo el el caso de la API de Google Maps. Ésta permite a desarrolladores usar sus productos, ya sea de manera gratuita o gracias a una licencia; dependiendo del uso [18].

[1] MuleSoft, "What is an API? (Application Programming Interface)". *Youtube*, 19 de junio de 2015 [Vídeo en línea] `https://www.mulesoft.com/resources/api/what-is-an-api` (último acceso 20 de junio de 2018)

[2] BBVA Open4U, "Qué es una API y qué puede hacer por mi negocio", 7 de junio de 2016. [En línea] Disponible en: `https://bbvaopen4u.com/es/actualidad/que-es-una-api-y-que-puede-hacer-por-mi-negocio` (último acceso 25 de junio de 2018)

- APIs de funciones en sistemas operativos

  Un ejemplo de sistema operativo que cuenta con una API de funciones es Windows 2000. Consiste en un conjunto de llamadas públicas al sistema para llevar a cabo distintas operaciones [19].

- APIs de servicios web o web API

  Estas interfaces conectan servicios web, entidades que proporcionan servicios a otras aplicaciones informáticas, entre sí. Se permite el acceso a un servicio concreto a través de peticiones. Existen dos arquitecturas distintas para definir cómo han de llevarse a cabo dichas peticiones. SOAP es un protocolo de intercambio de información basado en el mensaje, mientras que REST se centra en las estados y recursos, tal y como lo hace el protocolo HTTP (hace posible que se construyan sistemas de manera independiente a los datos transferidos [20]) [3].

  REST se caracteriza por una serie de normas que serán descritas a continuación dada su relevancia para el proyecto, y por operaciones estándar como GET y POST. SOAP tiene por lenguaje estándar al formato XML mientras que el intercambio de información en la filosofía REST se puede realizar tanto en el formato XML como en formato JSON [21]. Las figuras 3.1 y 3.2 muestran un ejemplo de formato XML y formato JSON respectivamente.

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

Figura 3.1: Ejemplo de formato XML [4]

[3] Germán Escobar, "¿Qué es una API?", 17 de septiembre de 2015. [En línea] Disponible en: `https://blog.makeitreal.camp/que-es-un-api/`

[4] Código obtenido en: w3schools.com, "JSON vs XML"[En línea]. Disponible en `https://www.w3schools.com/js/js_json_xml.asp` (último acceso 23 de agosto de 2018)

```
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

Figura 3.2: Ejemplo de formato JSON [4]

REST API

REST se puede entender como un estilo arquitectónico de software creado por el célebre científico de computación Roy T. Fielding en la disertación de su tesis doctoral. En una charla con Charles Severance sobre su tesis doctoral, R. T. Fielding explica que REST nació para estandarizar los protocolos web mediante una serie de restricciones. Dichas restricciones se crearon, en parte, para satisfacer las necesidades de empresas que comenzaban a involucrarse y querían usar la web como una de sus plataformas [22].

Las cinco restricciones obligatorias, previamente mencionadas, que delimitan el sistema REST según su creador, Roy T. Fielding, son las que se describen a continuación [23]:

- Cliente-servidor

  El cliente únicamente recibe los datos y el servidor solamente los envía. No es relevante para la otra parte cómo se han conseguido dichos datos o cómo se van a usar; son completamente independientes.

- Interfaz uniforme

  Con el fin de mantener la anteriormente mencionada individualidad cliente-servidor, la interfaz de la API debe permitir al cliente mandar una petición al servidor en un lenguaje independiente de los mismos.

- Sin estado

  Las peticiones al servidor son independientes; es decir, cada una de ellas contiene toda la información necesaria para ser procesada. Este punto elimina la posibilidad de errores parciales.

- Cacheable

  Con el fin de reducir el número de iteraciones, una REST API debe ser capaz de almacenar datos cacheables (para posibilitar la recuperación a gran velocidad de datos frecuente o recientemente usados).

- Sistema de capas

  La estructura de capas hace posible disponer de un tope a partir del cual el cliente no interacciona directamente con la arquitectura de software. Se consigue así incrementar el nivel de seguridad.

Finalmente, antes de proseguir con el desarrollo del presente capítulo, es fundamental explicar qué es Postman. La funcionalidad más relevante acerca de plataforma Postman para este trabajo es que permite realizar pruebas de peticiones o *requests* a APIs de manera sencilla. En lugar de implementar un código, se puede realizar una petición a una API tan solo indicando: el tipo de petición, la dirección web a la que se desea realizar la petición y qué tipo de autenticación requiere la API (ver apartado 3.1.2). Seguidamente, se debe rellenar los campos necesarios según el tipo de autenticación, el tipo de *request* y los requisitos de la API. Una vez realizada la petición, aparece la respuesta de la misma. Por último, el usuario puede obtener el código de dicha petición escrito en varios lenguajes de programación, entre ellos: Python [24] [5].

3.1.2. Autorización y autenticación

Al comenzar este apartado, es importante conocer la diferencia entre estos dos conceptos. Autorización significa comprobar si un usuario tiene acceso mientras que la autenticación se ocupa de verificar que el usuario es quién dice ser. A continuación, se explicarán los tipos de autenticación que puede requerir una API. Asimismo, se detallará el proceso OAuth 2.0 por su relevancia para el proyecto.

Tipos de autenticación

Según el tipo de API, los contenidos pueden estar más o menos restringidos; a continuación se explicarán diferentes niveles de autenticación de una API. Para la siguiente clasificación se han elegido los tipos de autenticación más populares entre los que Postman permite elegir.

[5] Postman, "Postman — Products". [En línea] Disponible en `https://www.getpostman.com/` (último acceso 13 de junio de 2018)

- *No Authentication*

  Existen APIs que, para acceder a cierto contenido público, no requieren autorización ninguna. Por ejemplo, la API gráfica de Facebook. [25].

- *Basic Authentication*

  Ésta es una forma de autenticación simple y poco segura dado que el nombre usuario y la contraseña son enviadas como texto. Para conseguir la autorización necesaria y realizar peticiones, el cliente debe enviar una *string* del tipo <nombre-de-usuario:contraseña>. Dicha *string* debe estar codificada en *base64*. Finalmente, la autorización se enviará de la siguiente manera: <*Authorization: Basic base64-string-codificada*>[26] [27].

- *Digest Authentication*

  Este tipo de autenticación mejora la seguridad puesto que esconde el nombre de usuario y la contraseña al encriptarlos [28].

- *Bearer Token*

  Como se explicará posteriormente, la forma de autenticación *Bearer Token* puede ser usada sola y funciona de un modo similar a *Basic Authentication*, aunque se creó como parte de OAuth 2.0. *Bearer* es una palabra inglesa que significa portador. Se puede deducir entonces que se proporcionará acceso a determinados contenidos únicamente al portador del *token*; sin requerir una prueba de la identidad del cliente [29].

  A diferencia de la autenticación *Basic Authentication*, el *token* no es creado por el usuario sino que debe pedirlo. Para ello, el cliente enviará una petición de *token* usando su usuario y contraseña y el servidor responderá con un *Access Token* y un *Refresh Token*. Después, el cliente enviará otra petición al servidor pero, esta vez, con el *token* que este último le ha proporcionado y pedirá el acceso a determinados contenidos. Seguidamente, el servidor decidirá si el cliente puede o no acceder a dichos contenidos [30].

- OAuth 1.0

  Este tipo de autenticación es similar al anterior pero más complejo. Se asemeja al método *Bearer Token* ya que se necesita varios pasos para conseguir las credenciales finales. OAuth 1.0 puede resumirse en tres pasos: solicitar un *request token*, obtener credenciales temporales e intercambiarlas por el *token* de acceso final.

  El proceso de autenticación OAuth 1.0 sigue los siguientes pasos [31]:

1. Para comenzar, el cliente debe tener una cuenta en una plataforma.
2. Asimismo debe registrar una aplicación en la página de su API (rellenar los campos: *consumer key*, *client secret* y *callback URL*.
3. Una vez verificados dichos campos, el servidor envía al usuario un *request token* junto con una *secret key*.
4. Posteriormente, el cliente es redirigido a la dirección de autenticación donde el usuario añade el *request token*.
5. Después, se pide al usuario que conceda el acceso al cliente. Seguidamente, el cliente es redirigido a la anteriormente mencionada *callback URL* donde aparece un *token* temporal y un *secret*.
6. A continuación, el cliente intercambia el *token* temporal por el *token* de acceso.
7. Finalmente, el cliente tiene acceso a los recursos protegidos de la página gracias a la API y al *access token*.

El diagrama de flujo de la figura 3.3 muestra un resumen de los anteriormente mencionados pasos para conseguir autenticación mediante el protocolo OAuth 1.0.

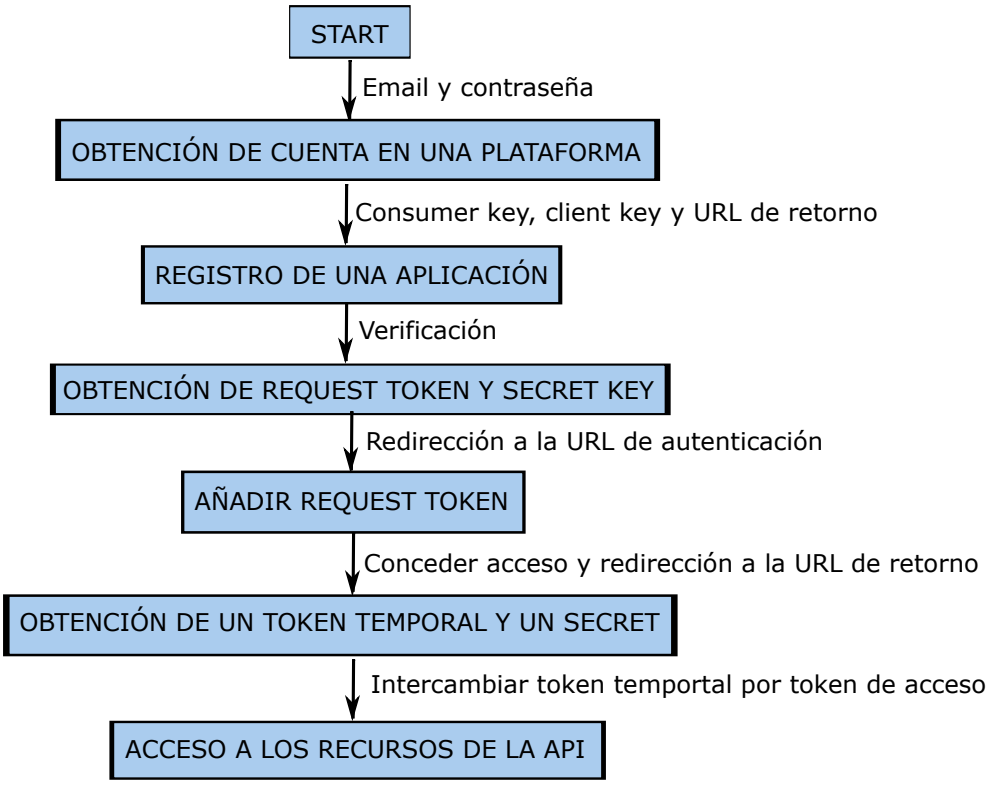

Figura 3.3: Diagrama de flujo de autenticación OAuth 1.0

- OAuth 2.0

  OAuth 2.0 es la versión más nueva de autenticación OAuth y la más usada por APIs como la de el conocido gestor de referencias, Mendeley. Su funcionamiento es más sencillo que el de OAuth 1.0 puesto que algunas páginas como Mendeley permiten un acceso directo y sencillo al *access token* (ver apartado 3.2.3). Resumidamente, el cliente debe registrar una aplicación para conseguir la siguiente información: *client secret, client ID, Authorization URL o endpoint, Access Token URL* o *endpoint* y *callback URL*. Con dicha información, puede solicitar el *access token* y, posteriormente, ejecutar la petición que desee a la página [27].

  En cierto modo, se podría encontrar una similitud entre OAuth 2.0 y *Basic Authentication* en cuanto que antes de acceder a la información, se requiere un paso previo para obtener las credenciales de acceso; el *access token*. Por otro lado OAuth 2.0 se puede comparar con la versión anterior, OAuth 1.0, ya que en ambos casos es necesario registrar un aplicación en la página web de la API. A continuación, se describirá con más detalle el proceso OAuth 2.0 y en el apartado 3.2.3 se explicará el proceso de autenticación OAuth 2.0 para la plataforma Mendeley.

OAuth 2.0

En este apartado se explicará cómo conseguir las credenciales necesarias para llevar a cabo una autenticación OAuth 2.0. Posteriormente (apartados 3.1.2 y 3.1.3), se detallará cómo usar dichas credenciales o *access token* para realizar una petición.

Una forma sencilla de conseguir el *access token* es a través de la plataforma Postman. En ella, basta con rellenar una serie de campos para obtener el *access token* directamente (ver apartado 3.2.3) y, después, elaborar la petición que se desee sin necesidad de escribir ningún código. Una vez hecha la petición, se puede obtener el código de dicha petición para JavaScript, Python y muchos otros lenguajes de programación. No obstante, antes de rellenar los campos, se debe registrar la aplicación en la página web de la API indicando el nombre y descripción de la aplicación, la dirección URL de retorno y el secreto del cliente.

A continuación, se describirán los requisitos necesarios para conseguir el *access token* y autenticarse a través del protocolo OAuth 2.0 basándose en [27] y [32]. Algunas páginas como la de Mendeley disponen de un modo particular para acceder al *access token*. No obstante, dado el carácter general de este apartado, se definirán los campos generales necesarios para seguir el protocolo OAuth 2.0. En el apartado 3.2.3, se desarrollará el otro camino que tienen los usuarios de Mendeley para conseguir las credenciales de autenticación dada su relevancia para el proyecto.

- *Grant Type*

  Se pueden distinguir cuatro tipos de concesión: *Authorization Code*, *Implicit*, *Resource Owner Password Credentials* y *Client Credentials*. El caso más general es *Authorization Code*. Éste pide rellenar todos los campos que se están describiendo en este apartado. Se obtiene al emplear como intermediario, entre cliente y dueño de recursos, un servidor de autorización. Las otras modalidades de concesión solo requieren algunos de los campos que se describen en este apartado.

  Estas últimas se diferencian de *Authorization Code grant* en que: *Implicit grant* no necesita de intermediario. Por otro lado, *Resource Owner Password Credentials grant* usa directamente el nombre de usuario y la contraseña para conseguir acceso. Finalmente, *Client Credentials grant* se suele emplear cuando el dueño de los recursos y el cliente son la misma persona; o si los recursos se encuentran bajo el control del cliente.

- *Callback URL*

  Esta URL ha de ser añadida por el cliente al registrar su aplicación; muchas plataformas proporcionan esta URL. El cliente es redirigido a esta URL en caso de que la autenticación sea correcta. De ella se obtiene el *access token*.

- *Auth URL*

  Esta dirección es de la que se obtiene el código de autorización.

- *Access Token URL*

  En esta URL ocurre el intercambio del código de autorización por el *access token*.

- *Client ID*

  El cliente, al resgistrar la aplicación en la página web de la API, recibe el identificador de cliente.

- *Client Secret*

  Como se ha explicado anteriormente, este un campo que el cliente debe rellenar al registrar la aplicación en la página de la API.

- *Scope*

  Indica el tipo de datos a los que se pide acceso. El valor de este campo suele ser *all* para no restringir el acceso a ningún recurso y ahorrar problemas futuros.

- *State*

  Valor empleado para evitar solicitudes de falsificación a través del sitio.

- *Client Authentication*

  Este campo ofrece dos opciones: enviar las credenciales en el encabezado o en el cuerpo. Esto depende de la API y algunas, incluso, admiten ambas formas enviar credenciales.

En la figura 3.4 se representa un resumen simple a modo de diagrama de flujo del proceso de autenticación OAuth 2.0.

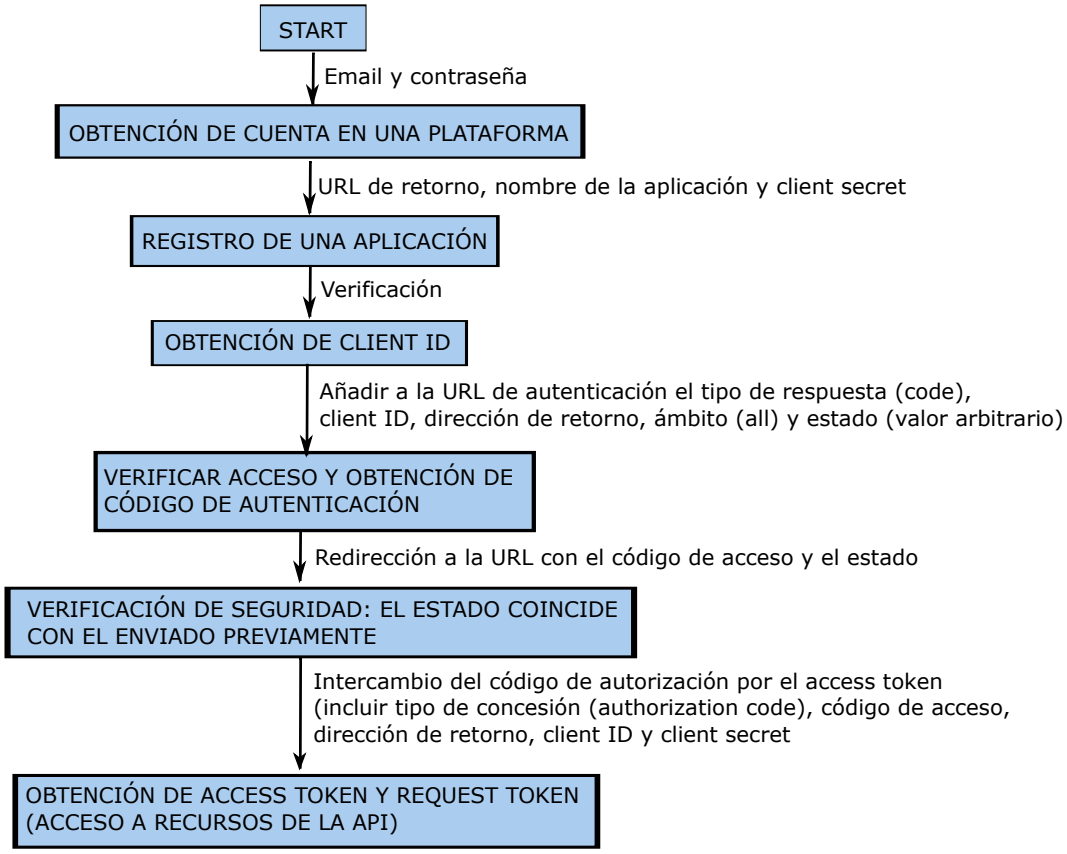

Figura 3.4: Diagrama de flujo del proceso de autenticación OAuth 2.0

3.1.3. *Requests* en Python

En este apartado se definirá el concepto de petición. Asimismo, se describirá brevemente cómo realizar peticiones con el lenguaje de programación Python.

Definición y elementos de una request

Una *request* es una petición que un cliente realiza a un servidor para conseguir acceso a recursos (p. ej. obtener, subir y modificar información). Se compone principalmente de los siguientes elementos [33]:

- URL

  Línea de texto que indica un recurso de Internet al que se desea acceder [34].

- Opciones o métodos de petición: verbos HTTP [35].

  También son conocidos como verbos de pedida e indican si lo que el cliente quiere es subir un archivo, eliminarlo, etc. Los más populares son los siguientes:

  - GET: se emplea únicamente para obtener información concreta de una página, como por ejemplo, descargar una lista de archivos o realizar una búsqueda.
  - HEAD: su funcionalidad es idéntica a la del método GET salvo por la respuesta. El servidor solo envía la información del encabezado y no la de todo el cuerpo de respuesta del documento.
  - DELETE: este método se emplea para borrar información como documentos de una página web.
  - POST: se utiliza para subir información al servidor, es decir, crear recursos. Se usa por ejemplo para subir un archivo PDF o una línea de texto a una plataforma.
  - PUT: crea un nuevo objeto o, si el elemento ya existe, modifica el recurso.
  - PATCH: es similar al método PUT aunque solo realiza modificaciones parciales.

- *Request header*

  Contiene información relativa al tipo de respuesta deseada, el nombre de un archivo que se va a subir, etc.

- *Request body*

  En esta parte se incluye la información que se desea enviar al servidor como una línea de texto o un archivo. Puede ser del tipo *data* o *file*, etc.

- *Response body*

  Contiene la respuesta a la pedida. Si el método de pedida es GET, entonces mostrará una lista con información sobre artículos, por ejemplo.

- *Response status-code*

  Su relevancia se debe a que, gracias a un número, el programador puede saber si su petición ha sido correcta y aceptada. También indica si, por el contrario, se le ha denegado la autorización o no es coherente la información enviada. Entre los códigos de estado más comunes se encuentran [36]:

  - 200 OK: respuesta satisfactoria tras una solicitud exitosa.
  - 201 *Created*: aparece tras una petición aceptada de POST o PUT indicando que se ha subido el archivo a la página web de la API a la que se ha hecho la petición.
  - 204 *No Content*: respuesta satisfactoria que, típicamente, tiene lugar tras una operación de DELETE.
  - 400 *Bad Request*: error de cliente debido a sintaxis no válida.
  - 401 *Unauthorized*: respuesta típica a un error del cliente tras enviar unas credenciales erróneas. Es muy común que ocurra al enviar un *token* anticuado en el caso de OAuth 2.0.
  - 403 *Forbidden*: este código indica un error del cliente al realizar una petición para la que no posee los permisos requeridos.
  - 404 *Not Found*: este error de cliente consiste en que el servidor no puede encontrar la respuesta demandada.

- 408 *Request Timeout*: error de cliente que ocurre cuando al servidor no recibe una petición en el tiempo que estaba preparado para esperar.

- 422: *Unprocessable Entity*: error de cliente en el cual la petición está bien formulada pero carece de semántica.

Ejemplo de request

En este apartado se mostrará un ejemplo de una petición por el método GET a la plataforma Mendeley en el editor PyCharm empleando el lenguaje de programación Python 3. El objetivo es descargar una lista de los diferentes tipos de identificadores de documentos con los que trabaja Mendeley (doi, isbn, etc). No se entrará en demasiado detalle para esta simple demostración. El apartado 3.2.3 habla en más detalle de la API de Mendeley, mientras que el apartado 4 es más técnico y en él se explicará todo lo necesario a cerca de la interacción con la API de Mendeley.

La figura 3.5 muestra el código de Python 3 que se debe ejecutar en el editor PyCharm para realizar la petición GET deseada. La figura 3.6 refleja la respuesta obtenida en la ventana "*Run*"de PyCharm. NOTA: el *access token* se obtiene con otro código pero de momento no es relevante (ver apartado 4.1).

```
import requests

url = 'https://api.mendeley.com/identifier_types'
headers = {'Authorization': 'Bearer ' + access_token.text}

response = requests.request('GET', url, headers = headers)
print(response.text)
```

Figura 3.5: Código para realizar una petición GET (identificadores de Mendeley)

```
[{"name":"arxiv","description":"arXiv ID"},{"name":"doi","description":"DOI
    "},{"name":"isbn","description":"ISBN"},{"name":"issn","description":"
    ISSN"},{"name":"pmid","description":"PubMed Unique Identifier (PMID)"
    },{"name":"scopus","description":"Scopus identifier (EID)"},{"name":"
    pui","description":"PUI"},{"name":"pii","description":"PII"},{"name":"
    sgr","description":"Scopus Group Identifier"}]
```

Figura 3.6: Respuesta obtenida en PyCharm al realizar una petición GET (identificadores de Mendeley)

A continuación se analizarán los elementos de la figura 3.5. La primera línea de código, *import requests* , sirve para importar el módulo de Python llamado *requests* que permite realizar peticiones. Seguidamente se encuentra el URL al que se va a realizar la petición. Éste está compuesto por dos partes: la primera, *https://api.mendeley.com/*, es común a todas las peticiones mientras que la segunda, *identifiers_types*, es específica para esta petición [6].

Una vez conocida la dirección URL a la que enviar la petición, es necesario enviar una autorización; tal y como se hace en el código. El formato es estándar y solo cambia el *token*, que es diferente para cada usuario y expira pasado un tiempo.

Finalmente, se realiza la petición gracias al módulo *requests*; se envía el verbo HTTP, la dirección URL a la que realizar la petición indicando qué se desea obtener y la autorización para ello en el encabezado. Se guarda la respuesta a dicha petición en una variable y, por último, se puede imprimir la respuesta añadiendo *.text* a dicha variable. Si se imprime solo la respuesta, lo que aparece en la pantalla es el código de estado de la respuesta (que en este caso es 200).

En la figura 3.6 aparece, como se esperaba, una lista de nombres y descripciones de los distintos tipos de identificadores de documentos con los que trabaja Mendeley. NOTA: Es posible modificar el código de la figura 3.5 de dos maneras sin alterar la respuesta (ver figura 3.7):

- La autorización, como se mencionó previamente, se puede enviar en el encabezado (figura 3.5) o en la propia URL (figura 3.7). Para enviarlo en la URL, simplemente se debe crear una *querystring* en la que asociar el valor del token con la clave *access_token*. Al realizar la petición, la querystring debe ser enviada en *params* como se muestra en la figura 3.7.

- La estructura del comando que envía la petición y guarda el valor de la respuesta en una variable puede ser modificada de la siguiente manera. En lugar de escribir *response.request("GET", url, ...)* (ver figura 3.5), se puede escribir lo siguiente *response.get(url, ...)* (ver figura 3.7).

[6] Mendeley, "Mendeley API Docs". 2015, [En línea] Disponible en: `https://api.mendeley.com/apidocs/docs` (último acceso 25 de agosto de 2018)

```
import requests

url = 'https://api.mendeley.com/identifier_types'

querystring = {'access_token': access_token.text}

response = requests.get(url, params = querystring)
print(response.text)
```

Figura 3.7: Código para realizar una petición GET (identificadores de Mendeley)

3.2. Gestores de referencias bibliográficas

En este apartado se definirá la función de un gestor de referencias bibliográfico y se describirán tres de los más populares entre los científicos e investigadores: ResearchGate, Google Scholar y Mendeley.

Se puede deducir por el nombre que un gestor de referencias bibliográficas es un software gracias al cual los usuarios pueden organizar, editar, almacenar y compartir referencias. Una referencia bibliográfica es el conjunto ordenado de la información necesaria para identificar una fuente de información; ya sea una página web, un vídeo, un libro o un capítulo de una revista.

Dependiendo del tipo de referencia y del estilo de la misma, los datos que especifican la fuente de información, el orden y la estética varían. Varía no solo la referencia en sí, sino también cómo aparecen a lo largo del documento cuando se las menciona. Por ejemplo, el estilo de referencias APA, muy empleado a nivel internacional en el ámbito de la educación, psicología y derecho, muestra las referencias en el texto de la siguiente manera *(Nombre, año, página)*. Por otro lado, en el estilo IEEE, empleado en documentos científicos, las referencias a parecen en la página con un número (*[1]*) [37].

3.2.1. ResearchGate

La plataforma ResearchGate ha sido, tal y como se define a sí misma, "creada por científicos, para científicos". Su principal objetivo es conseguir que las investigaciones científicas sean más accesibles [7]. A continuación se hará una descripción de las opciones que ofrece ResearchGate y se comentará brevemente qué tipo de API (ver apartado 3.1.1 para más información sobre qué es una API) tiene.

[7] ResearchGate, "ResearchGate: About" [En línea] Disponible en `https://www.researchgate.net/about` (último acceso 10 de agosto de 2018)

Funcionamiento de ResearchGate

Este popular gestor de referencias bibliográficas y documentos científicos ofrece varias opciones, expuestas a continuación, que atraen a miles de usuarios [38].

- Compartir publicaciones

  Desde la página principal se puede compartir una publicación haciendo clic en el botón azul "*Add new*" de la parte superior derecha. Se pide al usuario seleccionar el tipo de publicación que desea realizar. Después, debe rellenar los campos correspondientes (título, acceso privado o público, etc.). A continuación, puede agregar detalles como un resumen y añadir la publicación a un proyecto.

- Conectar y colaborar con otros investigadores

  Al subir una publicación, se puede añadir los colaboradores de dicha publicación; luego éstos verifican haber trabajado en la misma. Además, un usuario puede conectar con otros investigadores creando así una red de contactos. ResearchGate también dispone de una opción para enviar mensajes desde su página.

- Obtener estadísticas

  Una característica muy distintiva de ResearchGate es precisamente esta: proporcionar estadísticas. Un usuario puede ver cuantas veces otros usuarios han leído sus preguntas, publicaciones, etc. Estas estadísticas se muestran en una gráfica para poder observar la evolución a lo largo de los meses. No solo se generan estadísticas del número de veces que se ha leído una interacción del usuario (pregunta, publicación, etc.) sino que también disponen estadísticas de citaciones y recomendaciones.

- Formular preguntas y obtener respuestas

  Esta opción también es muy llamativa ya que otras páginas web del mismo sector no disponen de ella. El plantear preguntas o responder a las de otros usuarios fomenta el contacto entre ellos haciendo que puedan ayudar y ser ayudados. Además, se puede iniciar una discusión sobre el tema que desee el usuario.

- Encontrar el trabajo adecuado

  Gracias a una serie de filtros como la zona y el ámbito de trabajo; ResearchGate muestra las vacantes de trabajo disponibles. Asimismo, también se puede buscar un trabajo directamente sin necesitar los filtros.

API de Research Gate

Aunque muchas páginas web hoy en día ofrecen una API pública, ese no es el caso de ResearchGate. El popular gestor de referencias bibliográficas con aspectos de red social no dispone de una API abierta al público [39] [38]. Numerosos usuarios consideran la ausencia de una API accesible como gran desventaja frente a otros gestores que sí disponen de una.

3.2.2. Google Scholar

Esta plataforma se caracteriza por facilitar la citación, búsqueda y gestión de fuentes de información científicas (tesis, libros, conferencias, etc.) [8]. En este apartado se explicará cómo funciona Google Scholar y se hablará de su API.

Funcionamiento de Google Scholar

Basándose en la página oficial de Google Scholar [8] y en [40], se distinguen cinco funcionalidades que ofrece Google Scholar:

- Encontrar fuentes

  La página principal de Google Scholar es muy similar a la del famoso buscador, Google. El usuario puede introducir en el buscador una o varias palabras clave sobre las que desea encontrar información.

  Google Scholar propone como primer filtro el idioma de la fuente. También se puede filtrar por fecha y añadir citas y patentes a la búsqueda.

- Explorar publicaciones o autores relacionados

  Tras realizar una búsqueda, si el usuario encuentra un resultado que le interesa, puede seleccionar la opción "Artículos relacionados" y obtendrá una lista de éstos.

- Encontrar el documento completo

  Google Scholar muestra, en los resultados de la búsqueda, enlaces a los documentos enteros; también muestra más de una versión si existen varias. Si el documento se encentra en formato PDF, se mostrará, a la derecha del título, la opción de acceder directamente al mismo.

[8] Google Scholar, "About Us" [En línea] Disponible en `https://scholar.google.com/intl/es/scholar/about.html` (último acceso 10 de agosto de 2018)

- Citar y gestionar

  Bajo cada resultado de la búsqueda, aparecen dos iconos: una estrella y unas comillas. Una estrella que sirve para guardar los artículos deseados en "Mi biblioteca". Las comillas, por su parte, se emplean para citar el artículo en cuestión. Para ello, ofrece la posibilidad de copiar la referencia en varios estilos bibliográficos como archivos de BibTeX o EndNote, entre otros.

- Crear un perfil público y ver quién cita las publicaciones

  La opción "Mi perfil", brinda al usuario la oportunidad de crear un perfil. En él, puede indicar su nombre, afiliación, página web, etc. así como añadir artículos y configurar el perfil.

- Mantenerse al día en cualquier área

  En la parte inferior de la página de resultados, aparece la opción "Crear altera". Ésta propone al usuario recibir correos electrónicos con actualizaciones sobre una búsqueda determinada.

API de Google Scholar

Google Scholar, al igual que ResearchGate, tampoco ofrece una API pública a través de la cual interactuar con los contenidos de la página como se indica en [41]. No obstante, se proponen otros métodos para acceder y analizar el contenido de Google Scholar como el módulo de Python scholar.py [42]. Dicho módulo consigue resultados similares a los que se obtienen con una API. Por ejemplo, permite extraer datos relevantes acerca de una publicación como el título o el número de citaciones [42]. En la siguiente dirección se puede encontrar el módulo: scholar.py para hacer consultas a Google Scholar [9] .

3.2.3. Mendeley

Mendeley es uno de los gestores de referencias bibliográficas más populares entre investigadores y científicos debido a su comodidad. En este apartado se hablará de las opciones que ofrece esta plataforma; asimismo, se describirá la API de Mendeley.

[9] Página web para encontrar el módulo scholar.py: `https://github.com/ckreibich/scholar.py`

Funcionamiento de Mendeley

Mendeley, como otros gestores de referencias bibliográficos, dispone de una aplicación de escritorio en la que añadir una cuenta de usuario para que resulte más cómodo de usar. Gracias a dicha aplicación se pueden gestionar publicaciones y referencias así como leer y subrayar archivos PDFs, entre otras opciones; éstas se explican a continuación[43]. La plataforma Mendeley ofrece, entre otras, las siguientes funciones [44] [45]:

- Obtener referencias

  Gracias a una extensión del navegador que se puede descargar desde la aplicación de escritorio (*Tools ->Install Web Importer*), Mendeley facilita la importación de referencias. Esta extensión es muy útil puesto que, si un usuario encuentra un recurso que le parece interesante, puede clicar en el botón de la extensión de Mendeley y guardar la referencia. Una vez guardada la referencia, ésta aparece en la carpeta "*All Documents*". El usuario puede verificar que es correcta o editarla y después confirmar que es correcta.

- Citar

  Asimismo, Mendeley contribuye a citar referencias en los dos formatos de redacción más extendidos, LaTeX y Word. A continuación se explicará cómo.

  - LaTeX
    Las referencias de Mendeley se guardan en un archivo *.bib*. Dicho archivo puede ser importado a un editor de LaTeX como Overleaf, por ejemplo. Al añadir una referencia a Mendeley es necesario refrescar el archivo *.bib*.

    Después, empleando el comando adecuado para citar referencias en LaTeX, aparecen como sugerencia todas las claves o identificadores de citación del archivo de bibliografía. El usuario escoge la adecuada y continúa redactando. Al final del documento debe incluir la bibliografía en el estilo bibliográfico deseado con los comandos apropiados.

  - MS Word
    Mendeley ofrece otra herramienta, a parte de la que importa referencias de páginas web, para añadir referencias a archivos MS Word. Habilitando dicha herramienta (*MS Word Plugin*), se puede importar una cita de Mendeley haciendo clic en "*Insert citation*".

    Se puede buscar la cita por autor, título o año; o simplemente, acceder a la aplicación de escritorio y seleccionarla allí. Finalmente, con el botón "*Insert Bibliography*" se añade la cita en el estilo que se halla especificado en la línea de comandos de Word.

- Anotaciones

  Desde Mendeley se puede abrir archivos PDF y añadir anotaciones o subrayar sobre el mismo. Dichas anotaciones se pueden compartir con otros usuarios y no modifican el documento; se guardan en la cuenta. Sin embargo, se puede obtener un nuevo PDF con las anotaciones y subrayados con la opción "*Export with Annotations*".

- Organizar el contenido

  Con Mendeley se puede crear carpetas en las que organizar o clasificar los documentos y referencias bibliográficas según las preferencias del usuario. No solo se pueden gestionar y guardar en la aplicación de Mendeley, sino que también se pueden exportar al ordenador del usuario.

API de Mendeley

Una de las características más distintivas de Mendeley es su API y, especialmente, lo bien documentada que está (véase los recursos [6] y [10]). La API puede ser usada para numerosos fines, según lo que desee el programador. En este apartado, se hablará de la API de Mendeley y de cómo interactuar con ella. En la sección 4 se desarrollará en la práctica lo explicado en este apartado.

Características de la API de Mendeley

La API de Mendeley es una API de servicios web de tipo REST que sigue el protocolo OAuth 2.0 para la autenticación de peticiones y muestra las respuestas en formato JSON. Además, dispone de un módulo SDK para Python y JavaScript [46]. A continuación, se comentará cada una de esas características.

- API REST

  En el capítulo 3.1.1 se habla en profundidad de las REST API. En resumen, se trata de una serie de normas que sirven para proporcionar soporte de tareas como crear, leer, actualizar y eliminar datos. Entre dichas normas se encuentran tales como la independencia cliente-servidor, la uniformidad de la interfaz y la habilidad de almacenar datos cacheables [47].

[10] Mendeley, "API — Methods" [En línea] Disponible en: `https://dev.mendeley.com/methods/` (último acceso 25 de agosto de2018)

- OAuth 2.0

  El protocolo que emplea Mendeley para autenticar peticiones es OAuth 2.0. En el capítulo 3.1.2 se explica qué es y cómo puede un usuario autenticar una petición con este método. En pocas palabras, se trata de un método seguro que valida la petición si el servidor recibe el *token* correcto. Las credenciales del usuario nunca son expuestas, consiguiendo así proteger la identidad del cliente.

  NOTA: el *token* expira cada cierto tiempo y es necesario obtener uno nuevo. Posteriormente, en este capítulo, se mostrará cómo conseguir dicho *token*.

- Respuestas JSON

  JSON es un cómodo formato de texto en el que intercambiar información. Muestra la información de manera ramificada, lo que facilita su lectura [48]. Se muestra un ejemplo en la figura 3.2.

- SDK

  Un SDK es un conjunto herramientas que facilita la interacción de los programadores con una API. Mendeley ofrece un SDK para diferentes lenguajes de programación. El SDK para Python, por ejemplo, contiene numerosos archivos con extensión .py para ser usados directamente desde un editor de Python.

Obtención del *access token* y documentación relativa a peticiones

La interacción con la API se puede sintetizar en tres pasos: abrir, leer y escribir. O, en otras palabras, acceder a la cuenta (*log-in*), descargar información (*GET*) y subir información (*POST*). A su vez, estos tres pasos se pueden resumir en dos: conseguir el acceso y realizar el intercambio de información. A continuación se explicará cómo llevar a cabo estos dos pasos.

1. Conseguir el acceso (*log-in*)

   Existen varios modos de conseguir el *access token*; no obstante, todos ellos requieren el mismo requisito previo. Dicho requisito consiste en registrar una aplicación en la página web de la API: `https://dev.mendeley.com/`.

   Para registrar una aplicación, el usuario necesita antes introducir sus credenciales (email y contraseña). Posteriormente, debe indicar el nombre y descripción de la aplicación, un URL de re-dirección y generar un *client secret*. Seguidamente se asignará un identificador a la aplicación [11].

[11] Mendeley, "Hello Mendeley - Mendeley Developer Portal". [En línea] Disponible en: `https://dev.mendeley.com/getting_started/hello_mendeley.html` (último acceso 26 de agosto de 2018)

Por un lado, Mendeley dispone de una página web en la que obtener el *access token* directamente: `https://mendeley-show-me-access-tokens.herokuapp.com/` [11]. Introduciendo el correo electrónico y la contraseña, se obtiene el *access token* bajo el título *oAuth info*.

Por otro lado, existe otro modo de conseguir el *access token* con la anteriormente mencionada plataforma Postman. Para ello, es necesario rellenar los campos que se muestran en la figura 3.8. Los URLs se han obtenido en la siguiente página web [12].

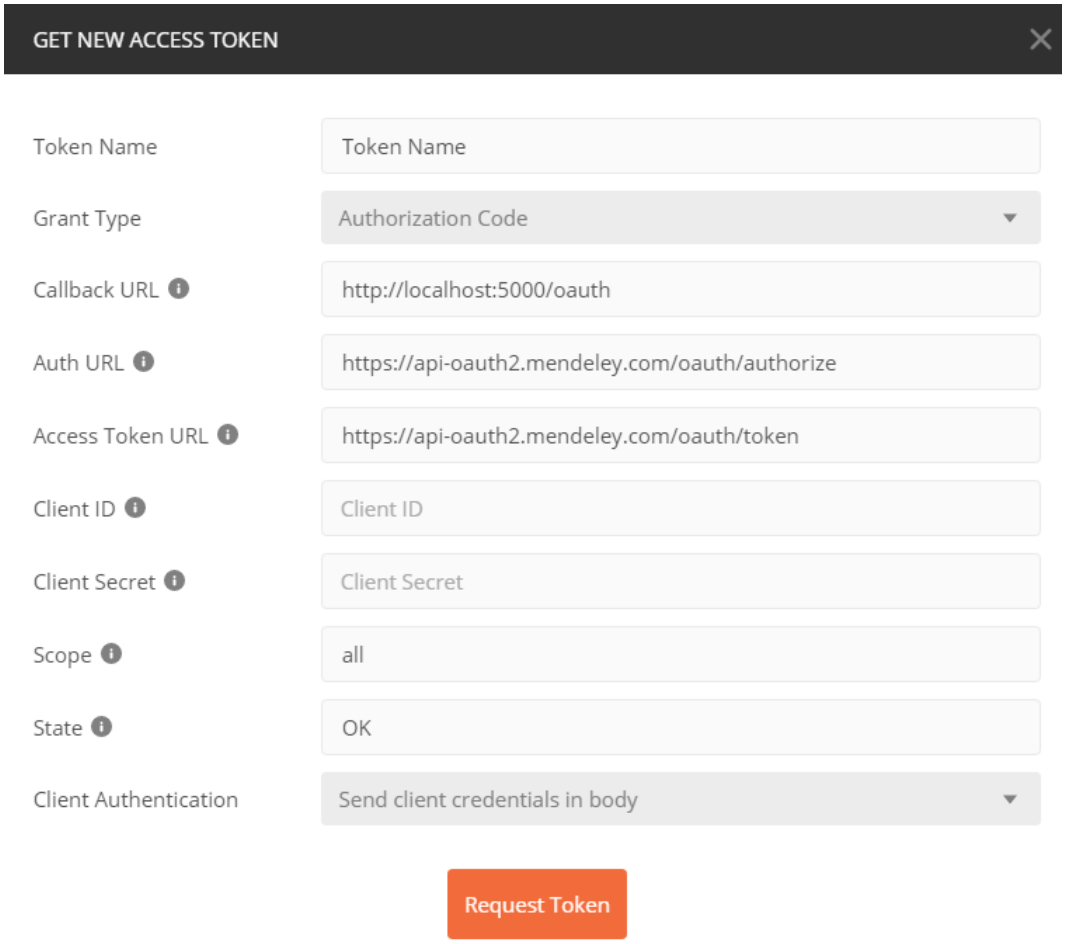

Figura 3.8: Obtención del *access token* mediante Postman

[12] Mendeley, "Authorization Code Flow - Mendeley Developer Portal". [En línea] Disponible en: `https://dev.mendeley.com/reference/topics/authorization_auth_code.html` (ultimo acceso 24 de agosto de 2018)

2. Intercambio de información (*GET* y *POST*)

   Mendeley dispone de dos páginas web en las que encontrar la documentación necesaria para hacer peticiones GET y POST (entre otras) [6] [10]. En ellas se muestra información relevante como el URL principal al que hacer peticiones y las extensiones necesarias según la petición (p. ej. *https://api.mendeley.com/search/documents*). También indican qué encabezados son necesarios y cuáles son los valores que se les debe. En el apartado 3.1.3 se mostró un pequeño ejemplo práctico de una petición GET.

Capítulo 4

Implementación

En este capítulo se detallará cómo se ha llevado a cabo la implementación de este proyecto. Ésta consta de dos partes: conseguir autorización y gestionar archivos (descargar una lista de archivos, subir y borrar documentos).

4.1. Proceso de autorización

La plataforma Mendeley cuenta con un proceso de autenticación llamado OAuth 2.0 (ver apartado 3.1.2 para más detalles). En pocas palabras, al realizar una petición a la API de Mendeley, es necesario enviar el *access token* en lugar de un nombre de usuario y contraseña.

Dichas credenciales se pueden conseguir gracias a una página web propia de Mendeley: `https://mendeley-show-me-access-tokens.herokuapp.com/` o en la plataforma Postman (ver apartados 3.2.3). No obstante, se optó por obtener el *token* a través de la previamente mencionada página web transformando los clics en código como se muestra a continuación.

Requisito previo: instalación de dependencias

Para poder pulsar botones y rellenar campos desde Python sin acceder directamente a una página web de un navegador, es necesario importar *webdriver* desde el módulo de Python Selenium (véase [49] para más información a cerca de este módulo). A continuación, se indica cómo instalar dicho módulo para los sistemas operativos Windows y Ubuntu.

- En Windows: simplemente descargar *ChromeDriver* en el siguiente enlace: `http://chromedriver.chromium.org/downloads`.

- En Ubuntu: para instalar las dependencias que precisa el código de este trabajo, se debe escribir en la terminal de Ubuntu lo que se muestra en la figura 4.1:

```
sudo apt install python3-selenium
sudo apt install python3-pypdf2
sudo apt install python3-bs4
CHROME_DRIVER_VERSION=`curl -sS chromedriver.storage.googleapis.com/
    LATEST_RELEASE`
wget -N http://chromedriver.storage.googleapis.com/$CHROME_DRIVER_VERSION/
    chromedriver_linux64.zip -P ~/
```

Figura 4.1: Código para descargar ChromeDriver en Ubuntu

Una vez instaladas las dependencias necesarias, se puede proceder a la programación del código en Python 3.7. Se usará el editor PyCharm 3.5 y el módulo Selenium. Para conseguir transformar la acción de pulsar un botón o copiar información a través del módulo Selenium, será necesario identificar dicho botón a través de su *XPath*. Por otro lado, para rellenar un campo en una página web de un navegador desde un código escrito en Python, se buscará dicho campo por su *id*. Más adelante se detalla cómo realizar lo que avanza este párrafo.

A continuación aparece una lista de pasos a seguir para conseguir la información necesaria al pulsar botones y rellenar campos de una página web desde un código de Python; es decir, sin acceder a ella directamente. Seguidamente se mostrará el código de Python implementado para conseguir el *access token* de Mendeley así como la respuesta que aparece en la PyCharm tras la ejecución del código.

1. **Acceder a la página inicial**

   Para comenzar la implementación y automatización del proceso de obtener el *access token*, es necesario acceder al enlace que se muestra a continuación desde cualquier navegador (ver figura 4.2):

   `https://mendeley-show-me-access-tokens.herokuapp.com/`

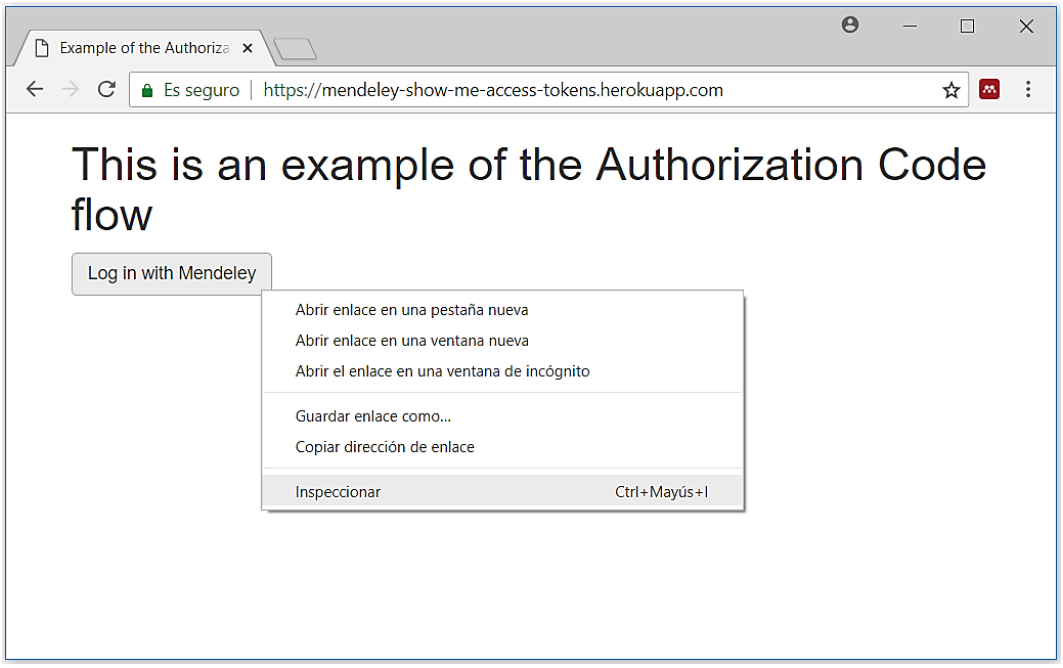

Figura 4.2: Página a través de la cual obtener el *access token*

2. **Acceder al panel de inspección de la página inicial**

   En segundo lugar, tal y como se muestra en la figura 4.2, se debe pulsar el botón derecho del ratón sobre "*Log in with Mendeley*". Después, hacer clic en "Inspeccionar". Seguidamente, aparece en la parte derecha de la pantalla el panel de inspección de elementos, como representa la figura 4.3.

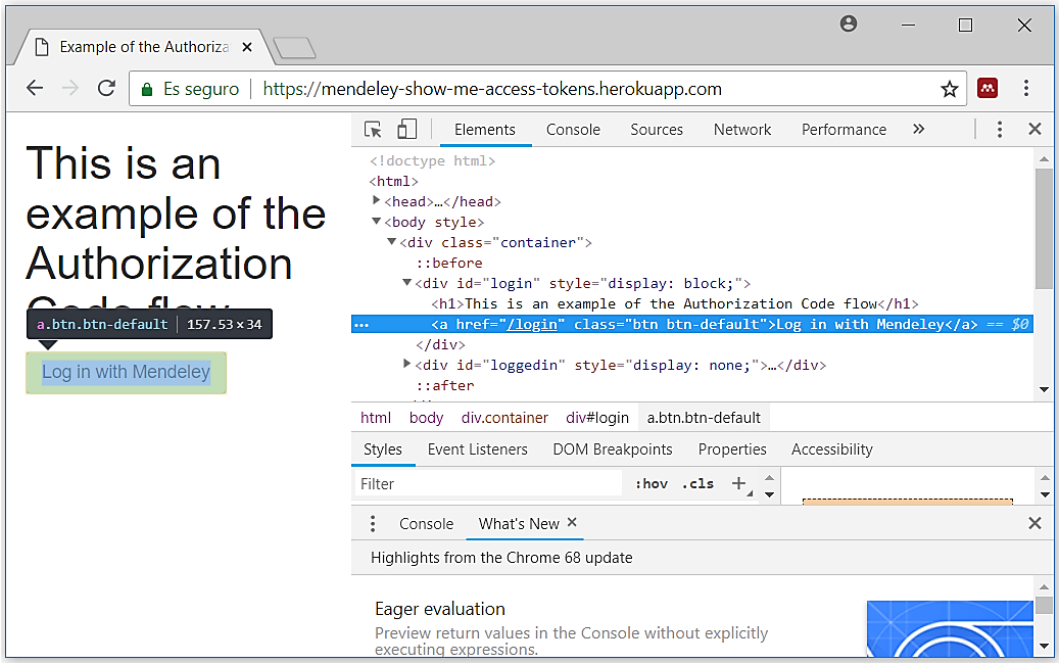

Figura 4.3: Inspección de elementos de la página inicial para obtener el *access token*

3. **Obtener el *XPath* de "*Log in with Mendeley*"**

   A continuación, se desea anotar su *XPath* del botón "*Log in with Mendeley*" para poder pulsarlo desde el código de Python.

   Para ello se debe: pulsar el botón derecho del ratón sobre la línea azul del panel de inspección que se muestra en la figura 4.3 (representa el elemento sobre el que queremos hacer pinchar). Del desplegable que aparece, se debe seleccionar la opción "*Copy*", y después "*Copy XPath*"; como se muestra en la figura 4.4.

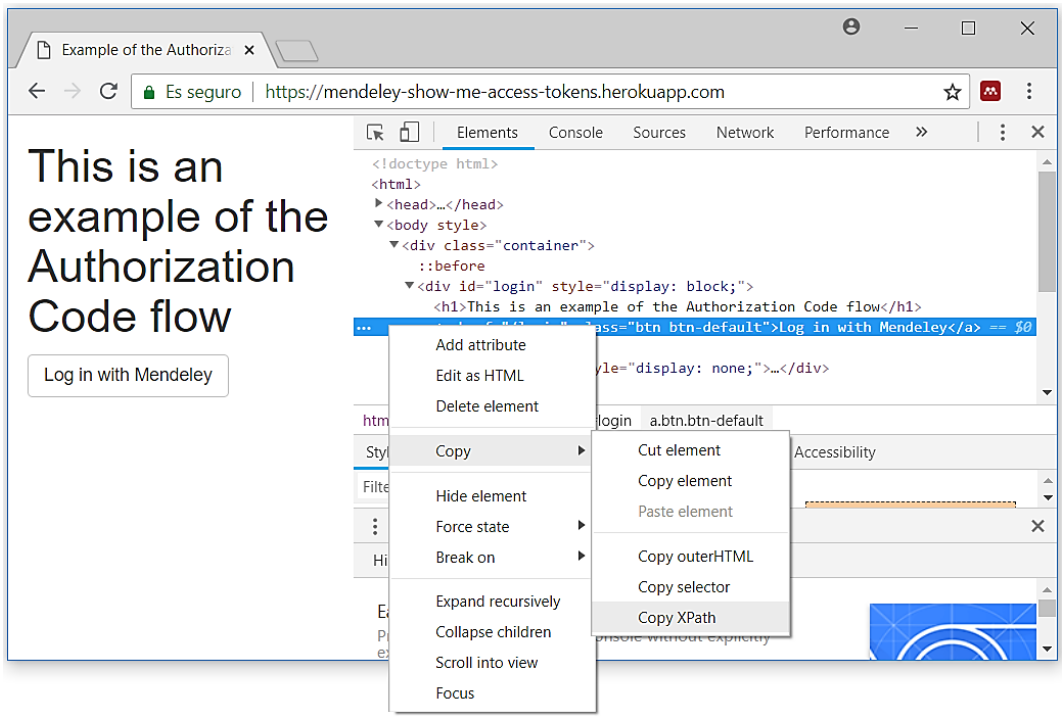

Figura 4.4: Obtención del *XPath* del elemento "*Log in with Mendeley*"

RESULTADO: El *XPath* del botón "*Log in with Mendeley*" es: //*[@id="login"]/a

4. **Acceder a la página de registro**

Tras pulsar manualmente el botón "*Log in with Mendeley*", aparece una página de registro. En ella, el usuario introducirá las credenciales con las que accede a su cuenta de Mendeley; es decir, su dirección de correo y su contraseña.

Antes de eso, debe inspeccionar el elemento. Es importante recordar que para inspeccionar un elemento es necesario situar el cursor del botón sobre él, pulsar el botón derecho del ratón y seleccionar la opción "Inspeccionar". La figura 4.5) recuerda cómo inspeccionar el nombre del campo "*Email*" .

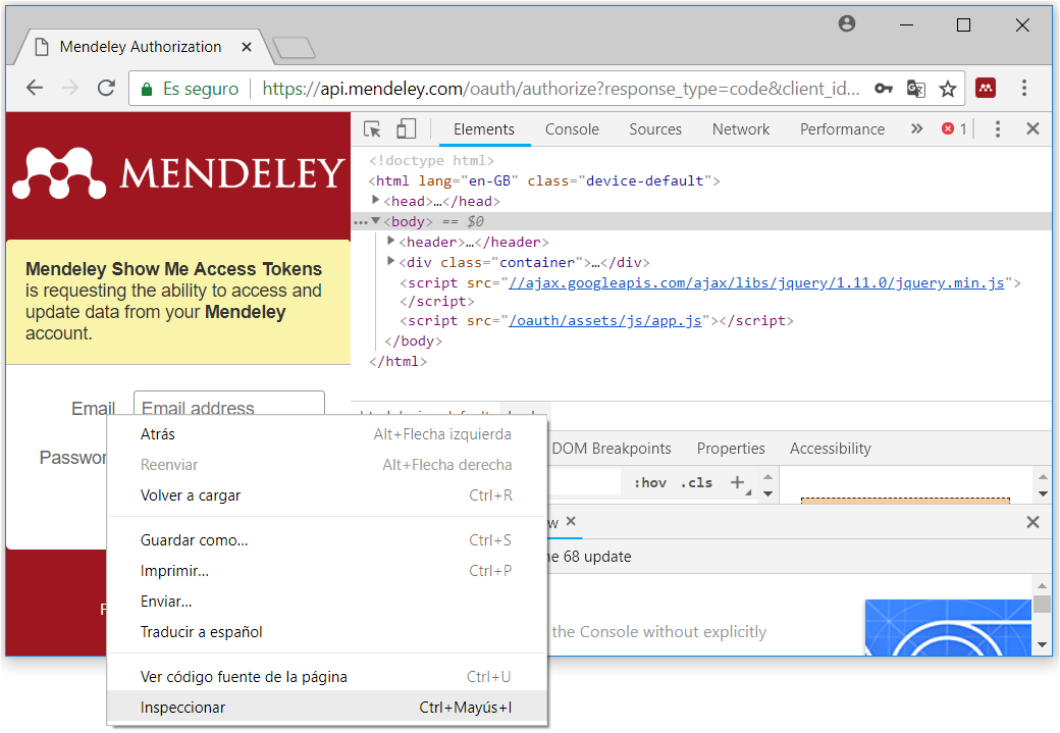

Figura 4.5: Página de registro

5. **Conseguir el *id* del campo "*Email*"**

   El acceso a los datos del campo "*Email*", se puede realizar de dos maneras distintas:

   a) Inspeccionando el nombre del campo

   Inspeccionar el nombre del campo, "*Email*" como se indica en el paso anterior (figura 4.5) y guardar la palabra que aparezca después de *label for* como se indica en la figura 4.6 al dibujar una elipse alrededor de la palabra.

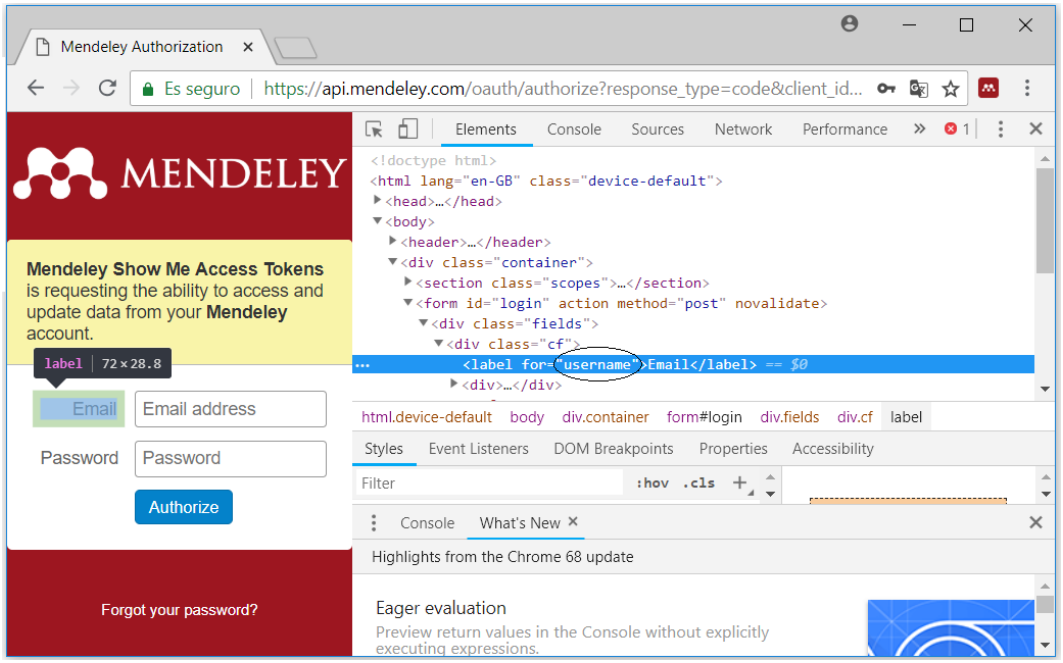

Figura 4.6: Conseguir el identificador del campo "*Email*" inspeccionando el nombre

b) Inspeccionando el campo en si
Tomar la *string* que sigue a *id* al inspeccionar el campo "*Email address*" como queda reflejado en la figura 4.7.

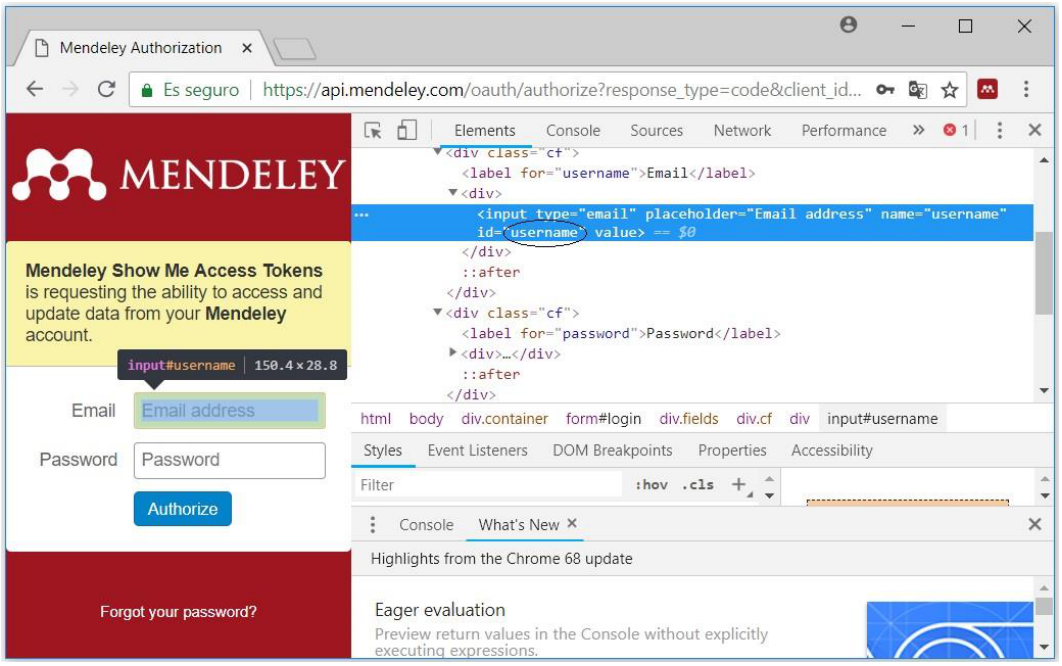

Figura 4.7: Conseguir el identificador del campo "*Email*" inspeccionando el campo

RESULTADO: El *id* para rellenar el campo "*Email*" empleando el módulo de Python, Selenium, es: *username*

6. **Conseguir el *id* del campo "*Password*"**

   Repetir el paso 5 pero para el campo "*Password*".

   RESULTADO: El *id* para rellenar el campo "*Email*" es: password.

7. **Obtener el *XPath* de "*Authorize*"**

   Repetir los pasos 2 y 3 para "*Authorize*" en lugar de "*Login with Mendeley*".

   RESULTADO: El *XPath* de "*Authorize*" es: //*[@id="login"]/div[2]/button.

8. **Acceder al *Access token***

   Una vez obtenida la información necesaria de los campos de registro, se puede cerrar el panel de inspección ("x" en la parte superior derecha); de esta manera se verá más cómodamente la siguiente página. Tras pulsar manualmente sobre el botón "*Authorize*", aparece una nueva página con información sobre el usuario, el *Access token* y *Refresh token* (ver figura 4.8).

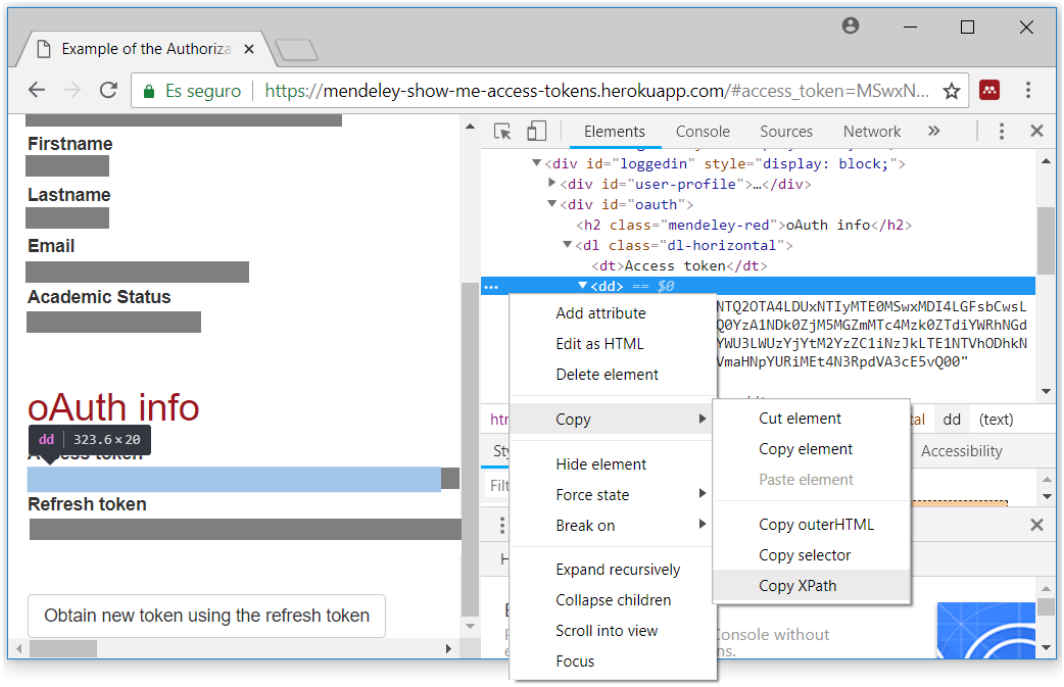

Figura 4.8: Página con información del usuario, *access token* y *refresh token*

9. **Obtener el *XPath* del *Access token***

   Repetir los pasos 2 y 3 para, posteriormente, guardar el valor del *access token* (ver figura 4.9).

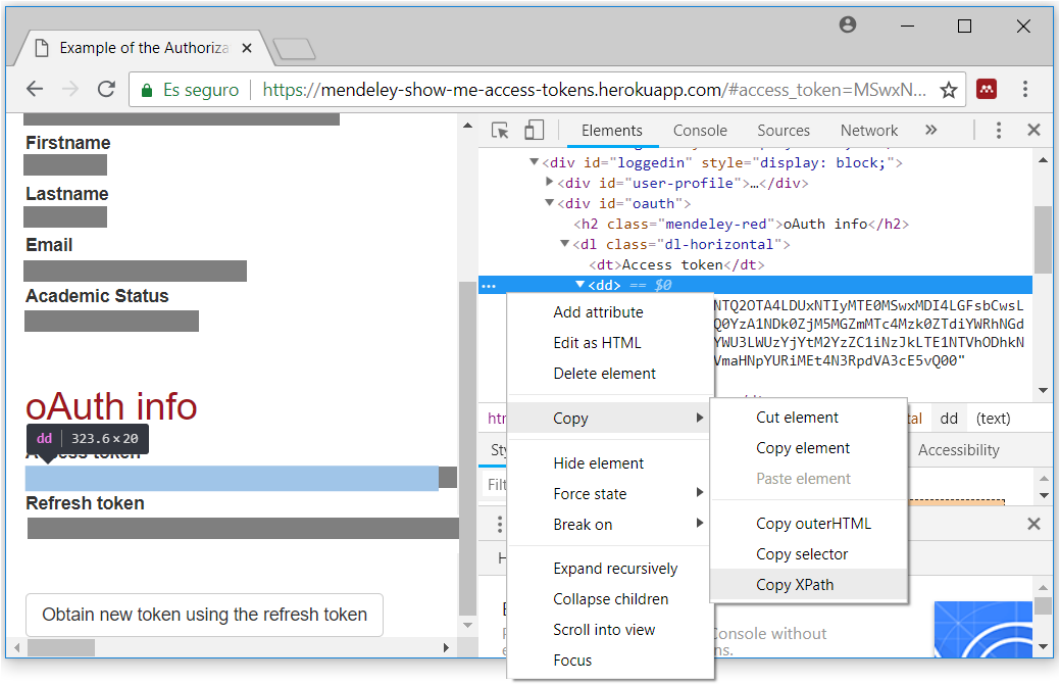

Figura 4.9: Obtener el *XPath* del valor del *Access token*

RESULTADO: El *XPath* del valor del *access token* es://*[@id=“oauth”]/dl/dd[1]

10. **Escribir el código en Python**

    Una vez obtenida la información necesaria para, gracias al módulo Selenium, pulsar botones y rellenar campos; se puede proceder a redactar el código. La figura 4.10 muestra el código, escrito en Python 3, necesario para obtener el *Access token* con el que acceder a la API de Mendeley. Las preguntas que el programa realiza al usuario para obtener sus datos así como la respuesta final (*Access token*) aparecen en la figura 4.11.

    NOTA: El código de la figura 4.10 debe incluirse siempre antes de los códigos del apartado 4.2. Sin él, el código dará error al no encontrar la variable *access_token*.

```
from selenium import webdriver
from time import sleep


chrome_path = 'C:\\Users\\<Nombre_del_usuario>\\Desktop\\chromedriver.exe'
driver = webdriver.Chrome(chrome_path)

url = 'https://mendeley-show-me-access-tokens.herokuapp.com/'
driver.get(url)
sleep(1)
driver.find_element_by_xpath('//*[@id="login"]/a').click()
sleep(1)

email = input('Introduzca su dirección de email de Mendeley: ')
password = input('Introduzca su contraseña de Mendeley: ')

driver.find_element_by_id('username').send_keys(email)
driver.find_element_by_id('password').send_keys(password)
driver.find_element_by_xpath('//*[@id="login"]/div[2]/button').click()
sleep(1)

access_token = driver.find_element_by_xpath('//*[@id="oauth"]/dl/dd[1]')
print(access_token.text)
sleep(1)

driver.close()
```

Figura 4.10: Código para obtener el *Access token* de la API de Mendeley desde Python

```
Introduzca su dirección de email de Mendeley: nombre.apellido@gmail.com
Introduzca su contraseña de Mendeley: contraseñaMendeley
Access token
```

Figura 4.11: Preguntas que el código de la figura 4.10 hace al usuario y *Access token* para interactuar con la API de Mendeley desde Python

Por motivos de seguridad en la imagen 4.11 no se incluyen ni la dirección email y contraseña usadas en el trabajo. Asimismo, tampoco se añade el *access token* obtenido puesto que, entre otros motivos, no serviría de nada usarlo; los token expiran y es necesario obtenerlos justo antes de usarlos.

4.2. Gestión de archivos

En este apartado se explicará cómo se ha conseguido gestionar archivos desde Python. El código final de este trabajo permite descargar un listado de referencias y documentos, subir documentos y eliminar documentos desde Python. Este apartado se basa en la documentación de Mendeley (véase [6] y [10]); y se apoya en el uso de Postman.

4.2.1. Descargar listado de publicaciones

Descargar un listado de publicaciones o, en general, hacer una *request* de tipo GET es relativamente sencillo. La figura 4.12 muestra el código para realizar una petición GET a la API de mendeley (*https://api.mendeley.com*) para descargar el listado de publicaciones (archivos y referencias) de la carpeta "*All Documents*". Para ello se realizará una petición a la extensión */documents* y se enviará la autorización en el encabezado "*Accept*" tal y como indica la documentación de la API de Mendeley [6] [10].

NOTA: La línea de código "*driver.close()*", debe aparecer una única vez y al final del código completo; para evitar así errores de código. Es decir, se debe eliminar dicha línea del código de la figura 4.10 al añadir el de la figura 4.12.

```
from selenium import webdriver
from time import sleep
import requests
import json

class color:
    BOLD = '\033[1m'
    END = '\033[0m'

url = "https://api.mendeley.com/documents"
headers = {
    'Authorization': "Bearer " + access_token.text,
}
list = requests.request("GET", url, headers=headers)

print(color.BOLD+'\nListado de publicaciones: '+color.END)
parsed = json.loads(list.text)
print(json.dumps(parsed, indent = 4, sort_keys=True))
```

Figura 4.12: Código para descargar listado de publicaciones de Mendeley

La figura 4.13 muestra un ejemplo de cómo aparece la respuesta al código de la figura 4.12. De cada publicación aparecen los datos que estén disponibles como el nombre de los autores, el año de publicación o el título.

```
El listado de referencias y documentos:
[
    {
        "authors": [
            {
                "first_name": "Tim",
                "last_name": "Berners-Lee"
            },
            {
                "first_name": "Roy",
                "last_name": "Fielding"
            },
            {
                "first_name": "Henrik",
                "last_name": "Frystyk"
            }
        ],
        "created": "fecha_y_hora_de_creación",
        "id": "identificador_de_documento",
        "last_modified": "fecha_y_hora_de_modificación",
        "profile_id": "identificador_de_perfil",
        "title": "Hypertext transfer protocol -- HTTP/1.0",
        "type": "report",
        "year": 1996
    },
    {
        "created": "fecha_y_hora_de_creación",
        "id": "identificador_de_documento",
        "last_modified": "fecha_y_hora_de_modificación",
        "profile_id": "identificador_de_perfil",
        "title": "EUR-Lex - l24216a - EN - EUR-Lex",
        "type": "web_page"
    }
]
```

Figura 4.13: Ejemplo de listado de publicaciones de Mendeley

Por motivos de privacidad, no se incluye en la figura 4.13 ni la fecha y hora de creación y modificación, ni el identificador del documento y del perfil de usuario.

4.2.2. Eliminar una publicación

La API de Mendeley permite eliminar una publicación, ya sea una referencia o un documento, a partir de su *id*. Dado que el programa debe interactuar con un usuario humano, no es adecuado pedirle el *id* de la publicación. El usuario puede no conocer el identificador de la publicación, pero sí debe conocer el título. Por esto, se decidió pedir al usuario el título de la publicación y conseguir que el código obtenga el identificador.

La figura 4.14 muestra el código, basado en la documentación de la API de Mendeley [6] [10], que se debe colocar debajo del código de la figura 4.10 para eliminar un archivo. Opcionalmente, si se desea ver el listado de publicaciones previamente, se debe colocar el código de la figura 4.12 entre los códigos mencionados anteriormente.

```
from selenium import webdriver
from time import sleep
import requests
import json

class color:
    BOLD = '\033[1m'
    END = '\033[0m'

delete = input('\nSi desea borrar alguna publicación '
               'introduzca el título de la misma'
               '\nSi no desea eliminar nada escriba: no'
               '\nRespuesta: ')
if delete == 'no':
    pass
else:
    select_id = delete
    div1 = list.text.split(select_id)[1]
    div2 = div1.split('id":"')[1]
    id = div2.partition('",')[0]
    print()
    print('ID: ', id)

    url = "https://api.mendeley.com/documents/" + id
    headers = {
        'Authorization': "Bearer " + access_token.text,
    }
    deleted = requests.request("DELETE", url, headers=headers)

    print(deleted)

driver.close()
```

Figura 4.14: Código para eliminar una publicación de la carpeta "*All documents*" de Mendeley

En la figura 4.15 aparece la respuesta obtenida al ejecutar el código de la figura 4.14. En ella se pide al usuario que introduzca el título de la publicación que desea borrar y se imprime el identificador a modo de comprobación. Asimismo, se imprime la respuesta: 204 (ver apartado 3.1.3) en caso de que la eliminación haya tenido éxito. De nuevo, en la figura 4.15 no se muestra el título real de un documento ni su identificador; el propósito de la imagen es únicamente mostrar la estructura de la respuesta.

```
1 Si desea borrar alguna publicación introduzca el título del mismo
2 Si no desea eliminar nada escriba: no
3 Respuesta: Título_del_documento
4
5 ID: identificador
6 <Response [204]>
```

Figura 4.15: Ejemplo de eliminación de una publicación de la carpeta "*All documents*" de Mendeley

4.2.3. Subir un documento

Este es el apartado que ha resultado más complejo y problemático a lo largo de la implementación de este trabajo. A continuación se describirá el obstáculo de este apartado y se explicará cómo subir un documento a Mendeley.

Obstáculos para subir referencias a través de la API

Mendeley permite subir documentos de diversas extensiones e importar referencias desde un navegador. Este gestor de referencias también facilita otra manera de añadir referencias: adjuntar un archivo de extensión *.bib*.

La aplicación es capaz de descifrar el archivo y extraer los datos para así transformarlos en referencias que aparecen en la carpeta "*All documents*". Por otro lado, la documentación de la API de Mendeley [6] [10] indica cómo subir archivos de extensiones *.bib*, *.txt* y *.pdf*. No obstante, lo que la API permite es "subir documentos"; es decir, al subir un archivo tipo BibTeX, lo sube como si fuera un documento más y no extrae las referencias del mismo.

Se intentó solventar el problema de que, al subir archivos *.bib* a través de la API no se extraigan las referencias, buscando un modo de subir los datos de una referencia directamente. No obstante, en la documentación de la API no aparece ninguna información que pueda resultar útil para llevarlo a cabo. Asimismo, el módulo SDK que ofrece Mendeley para Python tampoco documenta de manera clara cómo subir una referencia.

Implementación para subir un archivo a Mendeley

A continuación, se explicará cómo subir un documento BibTex, de texto o pdf a la carpeta "*All Documents*" de Mendeley. Una vez más, este apartado se basa en la documentación ofrecida por Mendeley para interactuar con su API [6] [10] así como en las explicaciones de [50] acerca de Python.

Según la extensión del archivo, se debe procesarlo de manera distinta para acceder a su contenido. En primer lugar, se pide al usuario que introduzca la carpeta en la que se encuentra el documento así como el nombre del mismo y su extensión. Seguidamente, se extrae la extensión del archivo. Después, se procesa el archivo de acuerdo con el método propio según su extensión. Este procesamiento consigue que, al añadir un archivo en la *request*, se pueda llevar a cabo de una manera uniforme en una línea de código.

La figuras 4.16 y 4.16 muestran el código que se debe añadir después de código de la figura 4.10 para subir un archivo a Mendeley. Es importante recordar que la línea "*dirver.close()*", debe aparecer una única vez y al final del código completo.

```
from selenium import webdriver
from time import sleep
import requests
import json
import os
import PyPDF2
from pybtex.database.input import bibtex

folder = input('\nIntroduzca el nombre de la carpeta en'
                        ' la que se encuentra el archivo: ')
os.chdir(folder)
file_name = input('Introduzca el nombre del archivo '
                  'con su extensión: ')
ext = file_name[-3::]

if ext == 'txt':
    f = open(file_name, "r")
    f.close()
if ext == 'pdf':
    pdfFileObj = open(file_name, 'rb')
    pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
    pageObj = pdfReader.getPage(0)
    pdfFileObj.close()
if ext == 'bib':
    parser = bibtex.Parser()
    bibdata = parser.parse_file(file_name)
    for bib_id in bibdata.entries:
        entries = bibdata.entries[bib_id]
```

Figura 4.16: Código para subir un archivo a la carpeta "*All Documents*" (1/2)

```
url = "https://api.mendeley.com/documents"
headers = {
    'Content-Disposition': "attachment; filename=" +file_name,
    'Accept': "application/vnd.mendeley-document.1+json",
    'Authorization': 'Bearer ' + access_token.text,
    'Content-Type': 'text/plain',
    'Content-Length': '11509'
    }
files={'file': open(file_name,'rb')}
postfile = requests.post(url, headers=headers, files=files)

print(color.BOLD+'\nSe ha subido este archivo: '+color.END)
print(postfile.text)

driver.close()
```

Figura 4.17: Código para subir un archivo a la carpeta "*All Documents*" (2/2)

La figura 4.18 muestra lo que aparece en la ventana de ejecución de PyCharm. En ella aparecen las dos preguntas que el usuario debe responder: carpeta en la que se encuentra el archivo y nombre del archivo con extensión. Finalmente, aparece un mensaje con los datos del archivo que se acaba de subir. En la difura únicamente aparece un esquema de la respuesta sin datos reales.

```
Introduzca el nombre de la carpeta en la que se encuentra el archivo:
    carpeta
Introduzca el nombre del archivo con su extensión: Título.extensión

El siguiente archivo ha sido subido:
{"title":"Título","type":"journal","id":"identificador","created":"
    fecha_y_hora_de_la_creacción","file_attached":true,"profile_id":"
    identidicador_de_perfil","last_modified":"fecha_modificación"}
```

Figura 4.18: Ejemplo de subida de archivo a la carpeta *"All documents"* de Mendeley

Se puede acceder al código de este proyecto en la siguiente dirección:
`TFG_ConectividadMendeley-Python-/TFG_MireyaCesterodeDompablo__CODE`

Capítulo 5

Guía de uso

Este capítulo está dedicado a explicar a un usuario cómo funciona el código desarrollado en este proyecto. Se trata de un código escrito en el lenguaje de programación Python 3.7 en el editor PyCharm 3.5 desde Windows.

5.1. Requerimientos previos

Para poder ejecutar y hacer uso del código presentado en este proyecto, es fundamental que se cumplan los requisitos expuestos a continuación:

- Disponer de un ordenador portátil, o bien, de sobremesa. A la hora de ejecutar el código se debe poseer acceso a Internet.
- El ordenador debe tener instalado Python 3.7 así como el editor PyCharm *Community Edition* versión 3.5 de la compañía JetBrains o un editor de Python similar.
- Como se menciona en el apartado 4.1, se debe tener instaladas las dependencias necesarias para llevar a cabo el proceso de registro. En dicho apartado se explica cómo instalarlas.
- El usuario debe disponer de una cuenta en Mendeley (dirección email y contraseña) a la cual desee acceder para gestionar archivos.
- Para poder interactuar con la API de Mendeley, es necesario que el usuario haya registrado una aplicación en la página web de la misma. El apartado 3.2.3 describe la API de Mendeley y en él se explica cómo registrar una aplicación.
- Para poder ejecutar el programa es fundamental que el usuario disponga del código que se proporciona al final del capítulo 4.

5.2. Manual de uso

La interacción con la interfaz de este trabajo es muy simple e intuitiva. Consiste en responder a las preguntas que el programa hace al usuario en la ventana correspondiente a "*Run*" en PyCharm. Las figuras 5.1-5.2 que aparecen a lo largo de este apartado, muestran un ejemplo completo de las funcionalidades que ofrece el código desarrollado en este trabajo. A continuación se explicará detalladamente cada parte.

Para comenzar, el usuario debe abrir el editor PyCharm. En la barra de herramientas superior, debe seleccionar: "*File*" ->"*New Project*". Después, debe indicar tanto la carpeta en la que desee que se guarde el proyecto como el nombre del mismo. Una vez creado el proyecto, debajo de la antes mencionada barra de tareas, aparece el nombre de éste. Para crear una hoja de Python, el usuario debe seleccionar el nombre del proyecto con el botón derecho. Se abre un desplegable y de éste debe seleccionar "*New*" ->"*Python File*"; después debe escribir el nombre de la hoja. Seguidamente, se copia el código proporcionado en el Anexo. Finalmente, de la barra de herramientas se selecciona "*Run*" ->"*Run...*" y el nombre de la hoja de Python.

El programa comenzará pidiendo al usuario su dirección email y contraseña de su cuenta de Mendeley. De esta manera el programa obtendrá el *access token*; lo que le permitirá interactuar con la API y acceder a los contenidos de su cuenta. El código no tiene problemas con la expiración del *token* Seguidamente, se pide al usuario que indique lo que desea hacer; para ello le ofrecerá una lista con tres opciones. Para elegir una de ellas, deberá escribir el número de la misma (ver figura 5.1). Si se introduce una palabra, letra o número erróneo, se repetirá la pregunta.

```
1 Introduzca su dirección de email de Mendeley: nombre.apellido@gmail.com
2 Introduzca su contraseña de Mendeley: contraseñaMendeley
3
4  Qu é desea hacer? (Introduzca un número)
5    1. Ver el listado de referencias y documentos con posibilidad de borrar
       alguna/o
6    2. Subir un documento
7    3. Salir
8 Respuesta:
```

Figura 5.1: Petición de credenciales y menú de opciones

1. **Descargar el listado de referencias y documentos con posibilidad de borrar alguna/o**

   Esta opción publica una lista en formato JSON con los datos de todas las publicaciones que existen en la carpeta "*All documents*" de la cuenta de Mendeley del usuario (figura 5.2). Una vez más, por motivos de privacidad tan solo se muestra la estructura de la respuesta y algunos datos reales como título y autores.

```
Respuesta: 1

El listado de referencias y documentos es:
[
    {
        "authors": [
            {
                "first_name": "Tim",
                "last_name": "Berners-Lee"
            },
            {
                "first_name": "Roy",
                "last_name": "Fielding"
            },
            {
                "first_name": "Henrik",
                "last_name": "Frystyk"
            }
        ],
        "created": "fecha_y_hora_de_creación",
        "id": "identificador_de_documento",
        "last_modified": "fecha_y_hora_de_modificación",
        "profile_id": "identificador_de_perfil",,
        "title": "Hypertext transfer protocol—HTTP/1.0",
        "type": "report",
        "year": 1996
    },
    {
        "created": "fecha_y_hora_de_creación",
        "id": "identificador_de_documento",
        "last_modified": "fecha_y_hora_de_modificación",
        "profile_id": "identificador_de_perfil",
        "title": "EUR-Lex - l24216a - EN - EUR-Lex",
        "type": "web_page"
    }
]
```

Figura 5.2: Descarga de listado de publicaciones

Tras descargar el listado de publicaciones, se ofrece la posibilidad de eliminar una de ellas introduciendo el título de ésta (estructura de la respuesta en figura 5.3).

```
Si desea borrar alguna publicación introduzca el título de la misma
Si no desea eliminar nada escriba: no
Respuesta: Título_del_documento

ID:  identificador
<Response [204]>
```

Figura 5.3: Eliminación de una publicación

2. **Subir un documento**

   La segunda opción permite subir un documento. En el apartado 4.2.3 se explica los obstáculos encontrados en la implementación de esta parte así como las limitaciones de la misma. Este código permite subir archivos con extensiones *.bib*, *.txt* y *.pdf*. Para ello, el programa pregunta al usuario en qué carpeta se encuentra el archivo. Seguidamente, le pide escribir el nombre del documento incluyendo su extensión (ver figura 5.4).

   NOTA: Si el usuario no sabe exactamente cuál es la ubicación en la que se encuentra el archivo, previamente debe: colocar el cursor sobre el archivo, pulsar el botón derecho del ratón, seleccionar la última opción, "Propiedades". Se abre una ventana y en la sección "General" aparece información del archivo como su "Ubicación".

```
Respuesta: 2

Introduzca el nombre de la carpeta en la que se encuentra el archivo:
    carpeta
Introduzca el nombre del archivo con su extensión: Título.extensión

El siguiente archivo ha sido subido:
{"title":"Título","type":"journal","id":"identificador","created":"
    fecha_y_hora_de_la_creacción","file_attached":true,"profile_id":"
    identidicador_de_perfil","last_modified":"fecha_modificación"}
```

Figura 5.4: Subida de un documento

3. **Salir y cerrar**

   El código funciona de manera cíclica, es decir, una vez finalizada una acción vuelve al principio y pregunta al usuario qué desea hacer. Cuando aparezca esa pregunta, el usuario puede seleccionar la tercera opción para salir del programa (ver figura 5.5). Después se cierra la página de donde se obtiene el *access token*.

```
Respuesta: 3

Process finished with exit code 0
```

Figura 5.5: Fin del programa

La figura 5.6 muestra un diagrama de bloques a modo de resumen del funcionamiento del código.

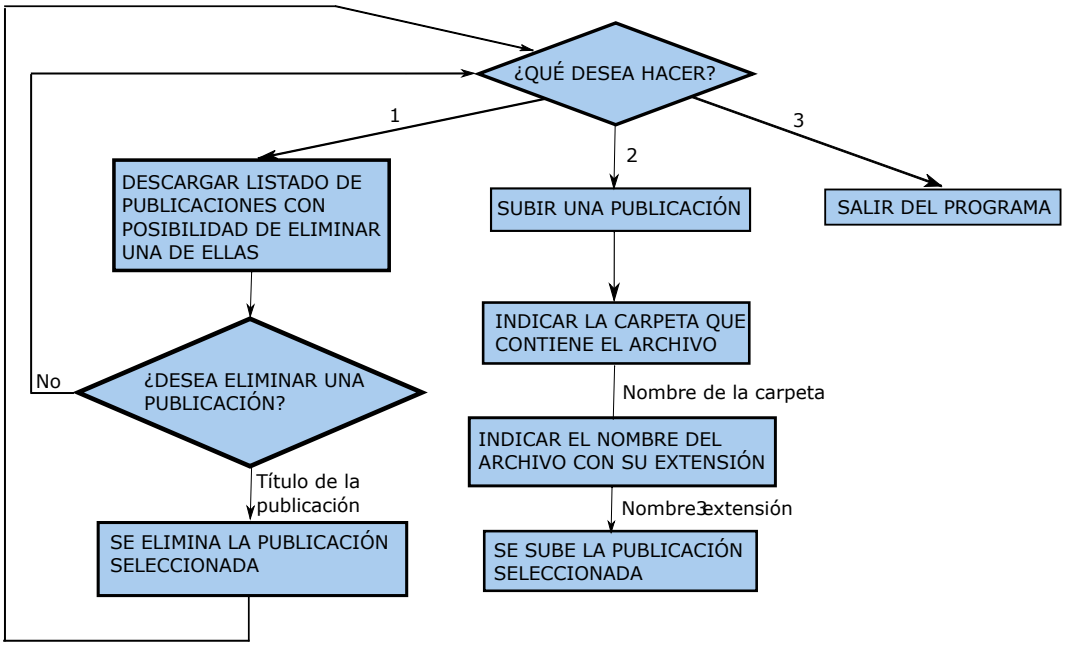

Figura 5.6: Diagrama de bloques del funcionamiento del código

Capítulo 6

Conclusiones

En este último capítulo, se procederá a resumir lo desarrollado en esta memoria. Asimismo, se llevará a cabo un análisis de los resultados obtenidos de este trabajo. Finalmente, se expondrá las líneas futuras que se esperan de eéste.

6.1. Conclusiones

Como conclusión, se realizará un resumen del contenido teórico del trabajo (apartados 1, 2 y 3) y un análisis de los resultados tras la implementación (4 y 5).

Resumen del contenido

El presente trabajo forma parte de un proyecto de posible relevancia a nivel internacional en el mundo científico. El planteamiento consiste en desarrollar una aplicación o plataforma que aúne varios gestores de referencias bibliográficas. En otras palabras, se desea poder acceder desde una única aplicación a los documentos de varias cuentas como Mendeley o ResearchGate. El fin último es aportar comodidad a los investigadores a la hora de gestionar archivos. Gracias a esta plataforma podrán compartir publicaciones en varias páginas web sin necesidad de acceder a cada una de ellas individualmente.

Dado que el trabajo lidia con la gestión de documentos de una cuenta y la obtención y procesamiento de credenciales, fue preciso estudiar el marco regulador legal en cuanto a protección de datos. Las normativas, tanto españolas y europeas, relacionadas con la protección de datos y el procesamiento informático, establecen que estos han de ser usados con consentimiento, de manera leal y no se permite su divulgación. Asimismo, en cuanto a los aspectos socio-económicos, las tecnologías web como las APIs suponen un gran avance en la obtención y procesamiento de datos. El aspecto de red social con el que cuentan algunos gestores de referencias los hace más atractivos para los investigadores. En cuanto al presupuesto, cabe mencionar los costes de personal: tutor y autora.

El cometido de este trabajo de fin de grado es estudiar gestores de referencias bibliográficas e iniciar dicho proyecto. En primer lugar, se observó el funcionamiento de diferentes gestores bibliográficos. En segundo lugar, se investigó cómo es posible acceder al contenido de una página sin entrar en ella directamente. Esta cuestión se encuentra resuelta, en la mayoría de los casos, por tecnologías web como las APIs. Éstas permiten, tras un proceso de autenticación, acceder y gestionar el contenido de una página web. En la actualidad son muy populares las APIs de tipo REST con protocolos de autenticación OAuth 2.0.

Si bien numerosas páginas web disponen de una API pública bien documentada y un módulo SDK para explorar el alcance de la misma, otras no la ofrecen. Durante el estudio de gestores de referencias, en concreto ResearchGate, Google Scholar y Mendeley, se observó que la oferta de APIs públicas en gestores de referencias no es tan común como se esperaba. Es más, de los tres gestores mencionados, únicamente Mendeley dispone de una API pública. Se trata de una REST API con módulo SDK, respuestas de tipo JSON y protocolo de autenticación OAuth 2.0. No obstante, Mendeley cuenta con una página en la que obtener las credenciales de autenticación o *access token* directamente.

Análisis de resultados

Al comenzar el trabajo, se esperaba poder interactuar con el contenido de una cuenta de Mendeley desde un código de Python. Dicha interacción consistiría en: acceder a la cuenta, descargar una lista de publicaciones, eliminar una de ellas y subir documentos y referencias bibliográficas. Si bien los cuatro primeros objetivos se cumplieron con creces, ese no es el caso del último. Como se menciona en la memoria, la API de Mendeley no documenta cómo, o bien subir un archivo *.bib* y que aparezca la referencia en la cuenta, o bien subir una referencia directamente. Esto último supone un obstáculo que podría solucionarse con un estudio aún más profundo de la API, con métodos alternativos, o cuando Mendeley ofrezca dicha operación.

Exceptuando el previamente mencionado obstáculo, el código consigue lo esperado. El programa creado es capaz de, con una dirección email y una contraseña, conseguir el *access token*, autenticarse y acceder a la plataforma Mendeley sin hacerlo directamente. Asimismo, como se explica anteriormente, se consigue descargar listados, eliminar y subir archivos. No obstante, la relevancia de este trabajo se encuentra en que ha sentado la base de un proyecto más ambicioso. En el apartado 6.2 se hablará de lo que se espera conseguir una vez finalizado el proyecto.

6.2. Líneas futuras

Si bien los planes de futuro para este trabajo no están completamente delineados, a continuación se expondrá la idea de futuro que se tiene sobre él.

El objetivo final consiste en: crear una aplicación que permita gestionar archivos de varios gestores de referencias. Por ejemplo: se quiere que, tras seleccionar un documento, el usuario pueda pulsar sobre el nombre de los gestores de referencias en los que desee compartir dicho documento. La motivación es ahorrar tiempo del tedioso proceso de acceder a cada página y seguir cada protocolo para subir el mismo documento.

Para ello, será necesario hallar un procedimiento alternativo para acceder a una cuenta y a su contenido a partir de un código, sin utilizar una API. Esto será de gran utilidad para plataformas como ResearchGate o Google Scholar, puesto que no disponen de API pública documentada. Por otro lado, se puede expandir el horizonte de posibilidades y considerar el estudio de otros gestores bibliográficos como ORCID y Zotero. Una vez estudiados los gestores más populares, se decidirá cuántos y cuáles incluir en la nueva plataforma.

Después de desarrollar los códigos individualmente (con el mismo lenguaje de programación, a ser posible), será necesario unirlos en uno único. Con dicho código, el siguiente paso es crear una aplicación de escritorio; y, tal vez, también de móvil. En ella, el investigador habrá sincronizado la cuenta de cada gestor. Dicha sincronización (usuario y contraseña) eliminará la necesidad de registrarse cada vez. Una vez dentro, el usuario podrá acceder a sus carpetas y seleccionar los documentos que desee subir y a dónde subirlos.

Finalmente, se investigará cuál es la mejor manera de dar a conocer esta nueva aplicación y situarla en el mercado. Se prestará especial atención a la opinión de los usuarios acerca de la plataforma para asegurase de que su satisfacción es elevada.

Bibliografía

[1] Flavia Baladán and Jimena Hernández Varela. Intimidad y privacidad frente a la interceptación de las comunicaciones electrónicas. In *XVI Simposio Argentino de Informática y Derecho (SID 2016)-JAIIO 45 (Tres de Febrero, 2016)*, 2016.

[2] Óscar Alzaga Villaamil, Ignacio Gutiérrez Gutiérrez, Fernando Reviriego Picón, María Salvador Martínez, and Jorge Alguacil González-Aurioles. *Derecho político español Tomo II: Derechos fundamentales y Órganos del Estado: Según la Constitución de 1978.* Editorial Centro de Estudios Ramon Areces SA, 2017.

[3] Unión Europea. *Carta derechos fundamentales.* Oficina del Parlamento Europeo en España, 2003.

[4] Araceli Mangas Martín. *Carta de los Derechos Fundamentales de la Unión Europea: comentario artículo por artículo.* Fundacion BBVA, 2008.

[5] Carlos Ruiz Miguel. El derecho a la protección de los datos personales en la carta de derechos fundamentales de la unión europea: Análisis crítico. *Revista de Derecho Comunitario Europeo*, vol. 7, nº 14, pp. 7-43, 2003.

[6] Juan Carlos I Rey de España. Ley orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal. *Boletín Del Estado*, vol. 298, nº 2, pp. 43088-43099, 1999.

[7] Roberto Mayor Gómez. Contenido y novedades del reglamento general de protección de datos de la ue (reglamento ue 2016/679, de 27 de abril de 2016). *Gabilex: Revista del Gabinete Jurídico de Castilla-La Mancha*, nº 6, pp. 243-280, 2016.

[8] Pedro Alberto de Miguel Asensio. Competencia y derecho aplicable en el reglamento general sobre protección de datos de la unión europea. *Revista Española de Derecho Internacional*, vol. 69, nº 1, pp. 75-108, 2017.

[9] Gemma Minero Alejandre. Presente y futuro de la protección de datos personales. análisis normativo y jurisprudencial desde una perspectiva nacional y europea. *Anuario jurídico y económico escurialense*, (50):13–58, 2017.

[10] Pablo Lucas Murillo de la Cueva. "la construcción del derecho a la autodeterminación informativa". Ed: Centro de Estudios Políticos y Constitucionales, 1999.

[11] Ana Rosa González Murúa. El derecho a la intimidad, el derecho a la autodeterminación informativa y la lo 5/1992, de 29 de octubre, de regulación del tratamiento automatizado de datos personales. 1994.

[12] Montse Bonet, Marta Civil i Serra, and Montse Llinés. Una década de políticas de gestión del espectro radioeléctrico en la unión europea (1997-2007). análisis de las consultas públicas, el marco normativo y las prioridades estratégicas. *Observatorio*, vol. 2, n.º 4, 2008.

[13] Javier Álvarez Hernando. *Guía práctica sobre Protección de Datos: cuestiones y formularios (e-book)*. Lex Nova, 2011.

[14] Alberto Cerda Silva. El"nivel adecuado de protección"para las transferencias internacionales de datos personales desde la unión europea. *Revista de derecho (Valparaíso)*, (36):327–356, 2011.

[15] Carlos Oliva Marañón. Redes sociales y jóvenes: Una intimidad cuestionada en internet. *Aposta. Revista de Ciencias Sociales, nº 54*, 2012.

[16] Ricard Martínez Martínez. El derecho fundamental a la protección de datos: perspectivas. *IDP: revista de Internet, derecho y política= revista d'Internet, dret i política, nº 5, pp. 47-61*, 2007.

[17] David Crystal. *El lenguaje e Internet*. Ediciones AKAL, 2002.

[18] Rubén Alcaraz Martínez and Mireia Ribera Turró. Mapas digitales y aplicaciones basadas en la localización: mejoras en su accesibilidad para las personas ciegas. *No Solo Usabilidad, nº14*, 2015.

[19] Andrew S Tanenbaum. *Sistemas operativos modernos*. Pearson Educación, 2003.

[20] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. Technical report, 1999.

[21] A De la Cruz. *Una aproximación MDA para la conversión entre servicios web SOAP y RESTFUL.* Tesis Maestría en Sistemas Inteligentes, universidad complutense de Madrid, Septiembre, 2013.

[22] C. Severance. Roy T. Fielding: Understanding the REST style. *Computer*, vol. 48, nº 6, pp. 7-9, jun. 2015.

[23] Shreyas Cholia, David Skinner, and Joshua Boverhof. Newt: A restful service for building high performance computing web applications. In *2010 Gateway Computing Environments Workshop (GCE 2010)*. IEEE, pp. 1-11, 2010.

[24] Bao Vu et al. Developing pilot api application using node js: a case study of 1up media oy. 2016.

[25] Christian Reuter and Simon Scholl. Technical limitations for designing applications for social media. In *Mensch & Computer Workshopband*, pp. 131-139, 2014.

[26] John Franks, Phillip Hallam-Baker, Jeffrey Hostetler, Scott Lawrence, Paul Leach, Ari Luotonen, and Lawrence Stewart. Http authentication: Basic and digest access authentication. Technical report, 1999.

[27] Dick Hardt. RFC 6749: The OAuth 2.0 Authorization Framework, IETF. Technical report, 2012.

[28] Aki Niemi, Jari Arkko, and Vesa Torvinen. Hypertext transfer protocol (http) digest authentication using authentication and key agreement (aka). Technical report, 2002.

[29] Pili Hu, Ronghai Yang, Yue Li, and Wing Cheong Lau. Application impersonation: problems of oauth and api design in online social networks. In *Proceedings of the second ACM conference on Online social networks*. ACM, pp. 271-278, 2014.

[30] Inge Alexander Raknes and Lars Ailo Bongo. approved, 1 approved with reservations. 2018.

[31] Eran Hammer-Lahav. RFC 5849: The OAuth 1.0 Protocol, IETF. Technical report, 2010.

[32] Ryan Boyd. *Getting started with OAuth 2.0*. O'Reilly Media, Inc., 2012.

[33] Dilmerd Atencio Flores and David Mamani Machaca. Interconectividad basado en api rest en aplicaciones de la municipalidad provincial de lampa. 2017.

[34] Tim Berners-Lee, Larry Masinter, and Mark McCahill. Uniform resource locators (url). Technical report, 1994.

[35] Rakesh Vidya Chandra and Bala Subrahmanyam Varanasi. *Python requests essentials*. Packt Publishing Ltd, 2015.

[36] Tim Berners-Lee, Roy Fielding, and Henrik Frystyk. RFC 1945: Hypertext transfer protocol–http/1.0, Net Working Group. Technical report, 1996.

[37] Francisco Moreno, Norma Marthe, and Luis Alberto Rebolledo. *Cómo escribir textos académicos según normas internacionales: APA, IEEE, MLA, VANCOUVER e ICONTEC*. Universidad del Norte, 2010.

[38] Christian Pieter Hoffmann, Christoph Lutz, and Miriam Meckel. Impact factor 2.0: Applying social network analysis to scientific impact assessment. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. IEEE, 2014, 1576-1585.

[39] Zahra Hammook, Jelena V Misic, and Vojislav B Misic. Crawling researchgate. net to measure student/supervisor collaboration. In *GLOBECOM*, 2015, 1-6.

[40] Anthony Giddens. Agency, structure. In *Central problems in social theory*. Springer, 49-95, 1979.

[41] Madian Khabsa and C Lee Giles. The number of scholarly documents on the public web. *PloS one*, vol. 9, nº 5, 2014.

[42] Miguel Ángel Jiménez Zarzuelo et al. Análisis de comunidades científicas basadas en fuentes de datos online. Tesis doctoral, departamento de ingeniería informática, universidad autónoma de madrid, 2014.

[43] Zaugg Holt et al. Mendeley: Creating communities of scholarly inquiry through research collaboration. *TechTrends*, vol. 55, nº 1, 2011, pp. 32-36.

[44] Lyssania Macías Morales et al. Guía de uso del manejador de bibliografía: Mendeley. Departamento de Biología Evolutiva, Facultad de Ciencias, Universidad Nacional Autónoma de México, 2015.

[45] Arindam Basu. How to Write Using Rich Text Format and Markdown in Latex and Overleaf. Universidad de Canterbury, 2016.

[46] Munir Salman, Michael Fuchs Wilhelm Buechner, Binh Vu, Holger Brocks, Jana Becker, Dominic Heutelbeck, and Matthias Hemmje. Integrating scientific publication into an applied gaming ecosystem. *GSTF Journal on Computing (JoC)*, vol. 5, nº 1, 2018.

[47] Rasib Hassan Khan, Jukka Ylitalo, and Abu Shohel Ahmed. Openid authentication as a service in openstack. In *Information Assurance and Security (IAS), 2011 7th International Conference on*. IEEE, 372-377, 2011.

[48] Douglas Crockford. The application/json media type for javascript object notation (json). Technical report, 2006.

[49] Unmesh Gundecha. *Selenium Testing Tools Cookbook*. Packt Publishing Ltd, 2012.

[50] Mark Summerfield. *Programming in Python 3: a complete introduction to the Python language*. Addison-Wesley Professional, 2010.